2005-09

Automating case reports for the analysis of digital evidence

Cassidy, Regis H. Friend

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

AUTOMATING CASE REPORTS FOR THE ANALYSIS OF DIGITAL EVIDENCE

by

Regis H. Friend Cassidy

September 2005

Thesis Advisor:	Chris Eagle
Second Reader:	George W. Dinolt

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2005	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Automating Case Reports for the Analysis of Digital Evidence		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Friend Cassidy, Regis			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

The reporting process during computer analysis is critical in the practice of digital forensics. Case reports are used to review the process and results of an investigation and serve multiple purposes. The investigator may refer to these reports to monitor the progress of his analysis throughout the investigation. When acting as an expert witness, the investigator will refer to organized documentation to recall past analysis. A lot of time can elapse between the analysis and the actual testimony. Specific reports may also be used in court as visual aids. Not all cases make it to court, but corporate managers will still likely want to review a case report. Since digital forensics is a relatively new field and can have a high learning curve, reports may be used as a mechanism for sharing knowledge of digital forensic practices.

Existing open source forensics tools are an inexpensive alternative to commercial products, but lack the functionality to generate case reports. Open source tools are more likely to be accepted by the professional forensics community with this added capability. This thesis adds case report features to the Sleuth Kit and Autopsy Forensic Browser suite of tools, the premiere open-source forensics analysis software currently available.

14. SUBJECT TERMS Computer Forensics, Digital Forensics, Digital Evidence, Case Reports, Autopsy Forensic Browser			**15. NUMBER OF PAGES** 235
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

AUTOMATING CASE REPORTS FOR THE ANALYSIS OF DIGITAL EVIDENCE

Regis H. Friend Cassidy
Civilian, Federal Cyber Corps
B.S., New Mexico State University, 2003

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2005

Author: Regis H. Friend Cassidy

Approved by: Chris Eagle
Thesis Advisor

George W. Dinolt
Second Reader

Peter J. Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The reporting process during computer analysis is critical in the practice of digital forensics. Case reports are used to review the process and results of an investigation and serve multiple purposes. The investigator may refer to these reports to monitor the progress of his analysis throughout the investigation. When acting as an expert witness, the investigator will refer to organized documentation to recall past analysis. A lot of time can elapse between the analysis and the actual testimony. Specific reports may also be used in court as visual aids. Not all cases make it to court, but corporate managers will still likely want to review a case report. Since digital forensics is a relatively new field and can have a high learning curve, reports may be used as a mechanism for sharing knowledge of digital forensic practices.

Existing open source forensics tools are an inexpensive alternative to commercial products, but lack the functionality to generate case reports. Open source tools are more likely to be accepted by the professional forensics community with this added capability. This thesis adds case report features to the Sleuth Kit and Autopsy Forensic Browser suite of tools, the premiere open-source forensics analysis software currently available.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No.DUE-0114018. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

I would like to thank Robert L. Hutchinson of Sandia National Laboratories for sparking my interest in digital forensics and for providing the support to receive training. I would also like to thank Brain Carrier for developing the programs The Sleuth Kit and Autopsy the Forensic Browser, and for providing them as open source to the digital forensic community. Finally, I would like to thank my thesis advisors Chris Eagle and George W. Dinolt for giving me the opportunity to work on this thesis and taking the time for its review.

THIS PAGE INTENTIONALLY LEFT BLANK

I INTRODUCTION

"Technology will constantly raise the sophistication of attacks and countermeasures, but digital evidence will persist [Ref 1]."

Today we thrive in a world of information technology in which we rely on our computer systems for every day tasks. However, as a consequence of our dependence on computers we have begun to witness a rise in computer crimes. A computer crime is generically defined as a crime committed with the use of a computer, or a crime in which a computer is the victim [Ref 2]. An important step in the process of solving these crimes is the ability for the examiner to properly document the course of action in an investigation and to provide a final case report. This thesis explores methods for improving the automation of case reports.

Computer crimes encompass a variety of offenses such as cyber attacks, terrorism, child pornography, gambling, money laundering, financial scams, extortion, sabotage, identify theft, and more. Automated attack tools have lead to an increase in cyber attacks [Ref 3, 4], and with dramatic increases in processor speeds, hard disk capacities and internet bandwidth, the job of the computer criminal easier. As the number of computer crimes grow, so will the need to identify and respond to these crimes. A proper response will involve an investigation to help understand the crime so that the perpetrators are caught, or at the very least, reoccurrences of the crime can be prevented.

Fortunately, computer crimes have a natural characteristic that is seen in every other type of crime. Where there is a crime committed there will be evidence left behind. It is up to forensic analysts to interpret facts from evidence to understand the nature of the crime and to aid in the conviction or acquittal of the suspects. To solve computer crimes that involve digital evidence a new

discipline of study in criminal justice has been adopted. This field is known as digital forensics.

Forensics is the application of science to the legal process [Ref 5] and therefore is subject to strict requirements. Digital forensics must be conducted accurately while maintaining the integrity of the original evidence. State Governments are starting to require forensic examiners to be certified professionals before digital evidence can be accepted into court [Ref 5]. At the very least, the forensic examiner is expected to document absolutely everything. Proper documentation that defends the practice and results of analysis procedures is essential for success in computer crime cases that make it to court [Ref 6]. The documentation for an investigation is often compiled into a case report. A complete and readable case report will benefit the investigators and other third parties interested in reviewing the case. Case reports may also serve to promote sharing of knowledge in investigative results and techniques among law enforcement communities.

A. DIGITAL FORENSICS BACKGROUND

Digital Forensics was defined by the First Annual Digital Forensics Research Workshop [Ref 7] as:

> The use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations.

Although described as a science in this definition, digital forensics is currently practiced as more of an ‘art’, lacking certain scientific disciplines that would make it a science [Ref 2, 8]. However, this is slowly changing as more forensic analysts are becoming certified professionals through adequate training. Also,

there are continued efforts to develop examination standards that will provide structure to the digital forensics process.

A few process models have been proposed for governing a digital investigation. These models are the Forensics Process Model [Ref 9], the Abstract Digital Forensics Model [Ref 10], the Integrated Digital Investigation Model [Ref 11] and the Enhanced Digital Investigation Process Model [Ref 12]. These models were formed with the common goal to define a digital forensics process that adheres to an accepted standard and encompasses all digital technologies.

These forensic process models have slight differences, but every model aims to enforce a minimum set of principle requirements for processing digital evidence. The scope of this thesis is confined to the requirements that satisfy:

1. Evidence Documentation
2. Evidence Reconstruction
3. Evidence Reporting

1. Evidence Documentation

Documentation is the process in which digital evidence is properly documented as it is found, collected, and processed. It may be described in these models as a separate phase [Ref 11, 12] or it may be integrated into the Collection and Examination phases [Ref 9]. Regardless, the documentation of evidence should be ongoing throughout all phases of the investigation that involve any kind of handling of evidence

2. Evidence Reconstruction

Reconstruction of evidence is a process that stems from the analysis performed and involves piecing together evidence to develop conclusions and theories. The evidence is also evaluated for its significance and value to the case. In these models, evidence reconstruction is described either as part of the Analysis phase [Ref 9, 10] or it is a substitute for the Analysis phase [Ref 11, 12].

With the latter, evidence analysis is adopted into the Search and Collection Phase and into the Reconstruction phase.

3. Evidence Reporting

At the end of an investigation some form of reporting or presenting of the evidence is expected. A report should outline and summarize the examination process as well as provide explanations for any conclusions that have been made. The presentation of evidence may occur in a court room or it may be requested from corporate management.

Ideally, a case report should cover of all three of these requirements. The case report is a means to provide a reference to all the documentation in a case and to the results of analysis performed by the investigators. The case report is essential in the Presentation or Reporting phase, a phase described in every digital forensic process model.

B. THE CASE REPORT

The reporting process is critical in the practice of digital forensics. Case reports assist in the review of the procedures and results of an investigation and can serve multiple purposes. One purpose of the case report is to provide quality assurance for the way a digital investigation was carried out. Quality assurance is achieved by reporting the following:

- Chain of Custody: The case report will indicate who was handling what evidence and when.
- Audits: The case report will include a time log of all activity, any tools used, and any commands issued including their arguments.

The case report will also include the method used to validate the integrity of the evidence.

- Case Documentation: The case report will include all investigator notes and their history of actions.

In addition to providing quality assurance for the digital forensic process the case report also serves to portray the results and findings from the digital analysis. Some common forms of presenting results from digital analysis are depicted below. These items are standard in current forensic applications that support case report generation. Tools that include various forms of case reporting include EnCase [Ref 13] by Guidance Software and the Forensic Tool Kit [Ref 14] by AccessData. The tools typically support:

- Reports: The case report will include technical reports of files, meta data, segments of disks, etc.
- Keyword Searches: The case report will include a list of keyword searches performed and the results form those searches.
- Timelines: The case report will provide timelines of file activity, recreating the events that led up to the crime.
- Image Gallery: The case report will include copies of interesting or illegal images identified by the investigator. A majority of computer crime cases making it into the courts involve the possession and viewing of images that depict child pornography [Ref 4] so this feature is often needed.
- Bookmarks: The case report will categorize and describe evidence based on labels or bookmark names the investigator assigns to different evidence items.

The items that make up the content of a case report, such as, documentation, logs, and analysis results, should be continually updated as the investigation progresses. Throughout the investigation, the investigator may

need to refer to a snapshot of the case report to monitor the progress of his analysis. Once the investigation is complete the investigator will still need to refer to a case report.

Often, there may be a long period of time between the completion of an investigation and when the investigator serves as an expert witness in a court of law. The investigator will need to refer to well organized documentation to recall the procedures and results of past analysis. Parts of the report may also be used in court testimony as visual aids.

Digital forensics is a new field of study and can have a high learning curve. Ideally, the case report will be used as a mechanism for sharing knowledge of digital forensic practices so that the learning process can be expedited. With research, automated case reports can be extended to formally model the experiences, lessons learned, and knowledge discovered during the digital forensics process that conforms to a particular standard [Ref 2].

C. PURPOSE OF STUDY

There is a need for case reports to be automated by digital forensics analysis tools so that investigators can focus on the analysis and spend less time trying to write documentation. An investigation may span multiple computers or pieces of media and may involve multiple investigators. Therefore, proper documentation can often be lengthy and complex. The results from analysis, notes, and logs are at risk of being improperly recorded if manual documentation is relied on. If a case makes it to court and there is insufficient documentation to accurately reconstruct the forensic process, then the evidence may be subject to question and the credibility of the expert witness will be compromised.

Law enforcement agencies may not have sufficient funds to purchase computers and software dedicated to forensic analysis. As a result, if expensive commercial tools are relied on there is the risk of un-licensed copies making it

into the labs that process digital evidence. If illegal practices such as this are discovered then cases may be lost [Ref 6]. Open source forensics tools offer an inexpensive alternative to commercial tools.

Currently, open source tools for digital forensics lack the functionality to generate case reports. The Sleuth Kit [Ref 15] and Autopsy Forensic Browser [Ref 16] applications offer great functionality for analyzing disk images, such as creating timelines of file activity, recovering deleted files and performing keyword searches. But it is left up to the investigator to generate a single document that can be used for summarizing and for reviewing the case. With the additional feature to create case reports, these open source tools are more likely to be accepted into the professional field of digital forensics and under funded agencies will not have to rely on the expensive alternatives.

D. THESIS ORGANIZATION

This thesis presents a working prototype for the automation of case reports using the Autopsy Forensic Browser. Chapter II provides an overview to the Sleuth Kit and Autopsy Forensic Browser, two popular open source forensic applications used for processing digital evidence. Chapter III describes the enhancements that were made to Autopsy's case management. These enhancements were added in order to better support the creation of case reports. The contents of an Autopsy case report are discussed in Chapter IV. The case report is divided into the following sections: Case Information, Case Audit and Analysis Information. Chapter IV concludes with suggestions that will assist in the generation of meaningful reports. Chapter V presents the conclusions based on the work in this thesis and provides suggestions for future work. Appendices at the end of this thesis include an example of a case report, the complete source code for the case report module, and segments of source code for the modified modules in Autopsy.

THIS PAGE INTENTIONALLY LEFT BLANK

II BACKGROUND

This chapter provides an overview of the open source applications that were extended to support the automation and generation of case reports for digital forensics. The fact that these applications can be extended, as they were for the research in this thesis, is a demonstration of one of the benefits of open source software. Some background knowledge of these programs is recommended and may be necessary before reading about the modifications made to them discussed in Chapter IV.

A. OPEN SOURCE VERSUS CLOSED-SOURCE

During the course of a digital forensics investigation, a wide variety of software, both open source and proprietary (closed source) is available to digital forensics analysts. Whether open source is better than closed-source in general, has been an ongoing argument for years, and is likely to remain an open debate for the foreseeable future. One should recognize and weigh the benefits of both of these types of software given a specific situation and problem to solve. For example, digital forensics researcher Brian Carrier provides an excellent argument for the use of open source tools in a legal setting.

> To be admissible in a United States court, evidence must be both relevant and reliable… The judge's responsibility is to determine whether the underlying methodology and technique used to identify the evidence was sound, and whether as a result, the evidence is reliable…
>
> By publishing source code through open source extraction tools, the digital forensic community can examine and validate the procedures used to produce digital evidence [Ref 17].

In order for the claims and assertions made by an expert witness in court to be valid, they must demonstrate that the techniques and methods used in their investigation are sound and do not result in evidence corruption. The investigator trusts that the data acquisition and analysis tools he uses will satisfy this requirement. With open source applications there is no “magic” on how they work because they can be picked apart and inspected by anyone who chooses to do so. Because open source forensic applications can be tested through the analysis of their source code, they make an excellent choice for investigators in the forensic community. A forensic application has to be tested to show there are no false negatives (the tool provides all available data from the input) and no false positives (the tool does not introduce new data to the output) [Ref 17]. Open source applications also offer the additional benefit of being free.

If the court systems of the United States are going to require that tested and validated applications are used for processing digital evidence (as they should), then an open source approach can help to meet such requirements. Closed-source applications rely on a verification process that involves rigorous testing, but it is likely not all test cases possible can be accounted for; in theory there is an unlimited number of test cases. The methodologies and algorithms of an open source tool are subject to full disclosure and can be verified through the analysis of the source code in addition to thorough testing. While an open source analysis of an application does not necessarily imply it is a better application, it can help to validate the application with a higher level of confidence (giving the jury no reasonable doubt).

The motivation behind this thesis stems from the fact that there are already excellent open source digital forensic applications in wide spread use today. However, investigators still have to use commercial tools for various features, not yet available in open source tools, and in some cases to appease courts that have not accepted open source tools as a reliable means for collecting evidence. It should be realized that no open source or commercial application for digital forensics is perfect today; they all have their own benefits

as well as their “broken” components. This thesis focuses on methods for improving open source digital forensic applications, and in particular the automation of case reports.

This chapter provides an overview of Brian Carrier’s The Sleuth Kit and Autopsy Forensic Browser. At the time of this writing, Sleuth Kit is in version 2.02 and Autopsy is in version 2.05. The following chapters will discuss the work that has been done to improve these two applications and ideas for future work.

B. THE SLEUTHKIT

One of the most well known and well respected open source, digital forensics applications is The Sleuth Kit (TSK) suite of analysis tools. The author of TSK is Brain Carrier, a popular researcher in digital forensics [Ref 18]. TSK is a set of analysis tools that can be divided into what Carrier describes as extraction tools and presentation tools [Ref 19]. Extraction tools process a disk and its file system to describe the data and file content. Extraction tools must accurately report the data and detail its physical and logical structure on the disk. A presentation tool may sort or apply some filter to that data to make the data analysis more manageable for analysis.

TSK has several extraction tools that can be categorized to operate on the data at different layers of abstraction. It is important to identify and verify a piece of evidence at all layers possible to support a claim or hypothesis regarding the evidence. Carrier defines these layers as the Data Unit Layer, the Meta Data Layer and the File Name Layer [Ref 20]. The concept of these layers is also strongly emphasized in the digital forensics course offered by the SANS Institute [Ref 21]. Investigators should understand what it means to analyze evidence at these different layers. As such, case reports should clearly demonstrate that the evidence has been analyzed at all layers that apply.

1. Data Unit Layer

The data unit layer is the lowest layer of the three being defined here. In actuality, the data unit layer is a layer above the sector layer, which in turn is a layer above the individual track of a disk, and so on. A data unit can be described as a grouping of sectors. When data is written to a disk it is written in “chunks”, the size of a data unit, at a time. These are commonly known as clusters on Windows file systems, and blocks or fragments on Unix file systems. The size of these data units is dependent on the type of file system and the size of the partition in question. Data units are sequentially numbered, but this numbering schema is implied and not physically stored at the data unit’s location. A data unit’s number is determined by the byte offset on the disk or image, divided by the data unit’s size for that file system. The table below describes a list of data unit layer tools in TSK.

dcat	Dumps the contents of a specified data unit or range of data units. Commonly used to recover deleted files or file fragments in unallocated space.
dls	Similar to *dcat*, but dumps unallocated data units by default. Commonly used to create a separate image file representing the unallocated space of a disk. Analysis can be performed on this separately.
dstat	Displays the statistics of a data unit such as if it is allocated or unallocated.
dcalc	Converts from a data unit number in unallocated space to the actual data unit represented on the entire disk or image.

Table 1. Data Unit Layer tools in TSK

2. Meta Data Layer

The abstract layer above the data units is the Meta Data Layer. Meta data stores the details about a file. Details include a meta data identification number, allocation status, number of links to the file, user and group IDs, timestamps and the sequence of data units that make up the file. Examples of meta data structures are directory entries on FAT systems [Ref 22], Master File Table

(MFT) records on NTFS systems [Ref 23], and inodes on Unix and Linux based systems [Ref 24]. This layer is extremely important to the investigator for identifying and recovering deleted files and for reconstructing a sequence of events. The table below describes a list of meta data layer tools in TSK.

icat	Dumps the data units of a file which is specified in the meta data. Used for recovering complete files even if fragmented.
ifind	Identifies the meta data structure of a given file name or data unit.
ils	Lists the meta data structures and their contents in a pipe delimited format. Intended for parsing by other TSK tools.
istat	Lists the details of a specified meta data structure in a human readable format.

Table 2. Meta Data Layer tools in TSK

3. File Name Layer

The File Name Layer provides a view of the data that users are most familiar with and comfortable operating in. At this layer the data is presented to the user with a readable file name and directory structure. Information about a file that is stored at this level, including the file name and associated meta data structure, is typically contained within the raw data of the parent directory. On most file systems, if a file is deleted the file name and meta data structure will be retained at this layer until overwritten. Some file systems such as FAT will store modified, access, or change (MAC) times at the file name layer as well. The table below describes a list of file name layer tools in TSK.

ffind	Will determine the file name of a file given the meta data structure.
fls	Lists the file names, allocated and deleted, of a specified directory.

Table 3. File Name Layer tools in TSK

In summary, the data unit layer represents the raw bytes of a file. The investigator uses this layer to identify evidence in unallocated space and to testify

where on disk a given file was physically located. The meta data layer is used by the investigator to determine a variety of timestamps associated with the file such as the last time the file was modified or accessed and other meta information describing that file. In the case of a deleted file, the meta data may still point to the exact data units that contain a file. This is useful for file recovery and often necessary for recovering fragmented files. The file name layer is used by the investigator to determine the actual file name associated with a given file and provides a manageable view of the directory structure of the disk.

If possible, the case report should detail a file identified as evidence at all layers. However, evidence may be recoverable at only one or two layers. For example, a meta data structure associated with a deleted file may have been reallocated to a new file. In this case, the investigator may recover the deleted file from the data unit layer, but will not be able to determine any timestamps associated with that file. If information about the file has been overwritten at the file name layer as well, then the investigator may not be able to determine the original file name. An understanding of these layers is essential for the discovery and explanation of evidence that has been deleted or is straggling in unallocated and swap space.

4. Presentation Tools

TSK also includes some presentation tools that arrange the output from the extraction tools in a way that is easy for the investigator to use for analysis. A presentation tool may operate at any of the layers described. Below is a list of presentation tools in TSK.

mactime	Will create a timeline of activity from the output from *ils* and *fls*.
sorter	Sorts and groups files found from *ils* and *fls* based on file type.

Table 4. Presentation tool in TSK

Presentation tools aid the investigator in his analysis and play a large role in case reports. A client, judge or jury will need to be presented with a view of the data that is easy to understand and verify. The next section describes the presentation tool Brain Carrier developed in order to support TSK. It is called the Autopsy Forensic Browser and offers an HTML based front-end to the command-line based tools of TSK. This tool will be referred to as Autopsy for the remainder of the thesis. Understanding the concepts of layering described in this current section is essential for using the features and navigation of Autopsy.

C. AUTOPSY FORENSIC BROWSER

Autopsy is a graphical front-end to TSK that gives it capabilities that are similar to commercial products such as Encase [Ref 13] and FTK [Ref 14].

1. Case Management

In addition to providing an interface to the evidence extraction tools of TSK, Autopsy provides a very important case management feature. Each case in Autopsy is organized by investigators and by hosts. Multiple investigators may be assigned to a single case and since Autopsy is based on a server/client model each investigator can work on the case remotely from their own work station. A single case can encompass multiple hosts that are under investigation, each with their own time zone and time skew setting. A host may contain multiple disk images and each image may be composed of one or more partitions. Disk images and partitions are managed with an image or volume ID number assigned by Autopsy when the image is initially added to the case. Figure 1 below, has been generated to represent a case's hierarchal structure.

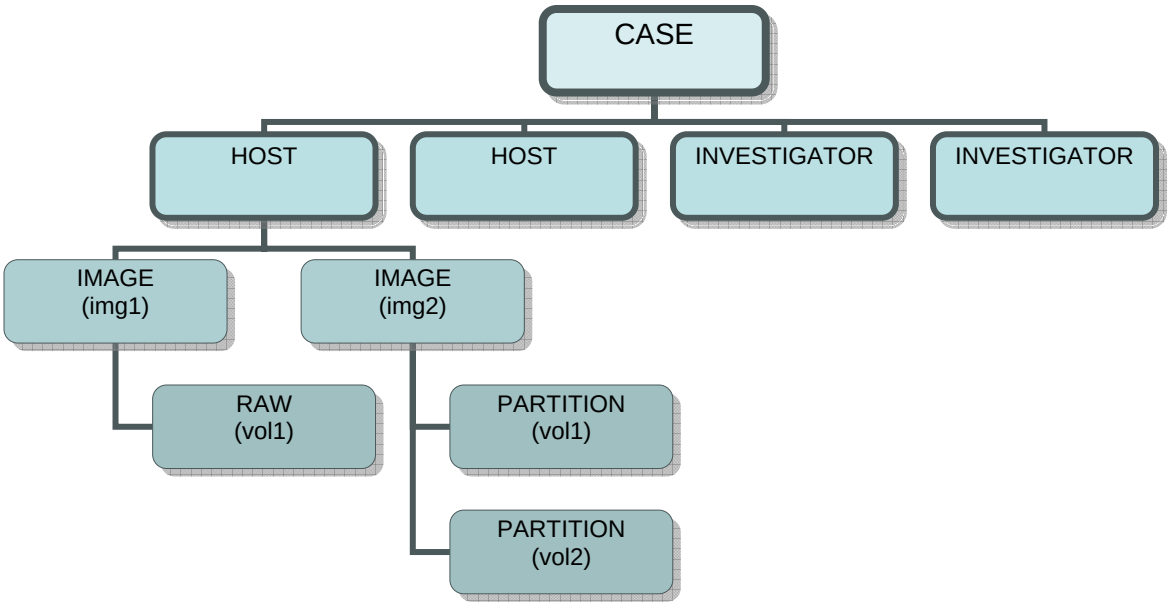


Figure 1. Case structure in Autopsy.

a. Case Auditing

Log files are generated for the individual case, host, and investigator. The case log records when the case was created, when a host is added, and each time the case is opened. Host log files contain similar information for each host, but also record activity indicating when each volume is accessed. The investigator logs contain the most information. Any action the investigator performs during his analysis is recorded, such as file listings, generation of timelines, keyword searching, etc. In addition to easily readable log files, a command log file is created for each investigator. This feature records TSK commands used to perform the investigator's analysis.

b. Notes and Sequencer Events

Throughout the analysis an investigator has an option to create notes. A note can be added for a file, a meta data structure, or a data unit. The purpose of creating a note is for the investigator to be able to quickly reference a piece of evidence to follow up on or to review at a later time. If the meta data structure of a file is still intact then the investigator will have the option to add the

note as a sequencer event for any combination of the three MAC times. Sequencer events are stored in separate files from notes and can be viewed in sorted order based on the time stamp. Sequencer events essentially create a time line of the suspect's actions on a given host and will be important to anyone who is reviewing the case.

c. Image Integrity

Computing hash values is a method used by investigators to verify that there was no evidence corruption during the course of analysis. When an image or volume is added to a case, an acquisition MD5 hash [Ref 25] is computed. At any time the investigator can compute the hash in Autopsy to verify the image's integrity. Autopsy is also able to compute and verify hash value for any additional images that are extracted from the original disk image such as, images of unallocated space, strings, and time line files.

d. Reports

Autopsy provides the investigator with the ability to create text-based reports, used for recording the details of a specified file or sequence of data units. These reports are separate and not the same as the case reports described in this thesis. Rather, the case report provides an overview of the entire case and will include these detailed reports for review. If a report is created for a file the following information is saved to the report:

General Information: This includes the file name, MD5 and SHA-1 hashes, name of the disk image, file system type, the date the report was generated and the investigator creating the report.

Meta Data Information: This information varies with the file system type, but will commonly include file attributes, file size, MAC times and a list of data units.

Content Information: The ASCII representation of the data content is saved in the report. To avoid a mess of symbols for non-ASCII characters in binary files, the output from running *strings* on the data can be saved instead.

Version Information: Shows the version of Sleuth Kit and Autopsy.

If a report is created for a sequence of Data Units then the Meta Data information is excluded from the report. The general information section will not include a file name but will include the data unit numbers and the size of a data unit in the file system. Additionally, the content section can be saved in a hex dump format. In addition to creating reports for files or a series of data units, the investigator can export MD5 values for all files in a specified directory.

2. Search and Analysis

Autopsy offers many search and analysis features for locating evidence. These features are implemented utilizing the Sleuth Kit, and other common Unix based tools.

a. File Listing

This feature provides a traditional view of the files on a volume that users are already familiar with. The investigator is able to browse through the directory and file structure of a suspects drive as if they were using the suspect's computer. This feature will display, if possible, any deleted files that may be present on the image. The primary TSK tool used in this feature is *fls*.

b. File Content

Autopsy allows the investigator to view a file or data in its raw, ASCII, strings or hex format. Viewing a file in its raw format is useful for file formats like HTML and graphical images. ASCII, strings and hex views are useful

for identifying suspicious strings in binary files. The primary TSK tools for listing a file's content are *dcat* and *icat*.

c. Hash Databases

The investigator may choose to compare the hash value of a given file with a known good or bad hash database. Autopsy supports the NIST National Software Reference Library (NSRL) [Ref 26] and investigators may also use their own index of known hashes that they have compiled. An investigator may look up any given file in a hash database to determine if it has already been flagged as malicious or ignorable. The TSK tool for looking up a hash value in a database is *hfind*.

d. File Type Sorting

Autopsy attempts to categorize the files on an image based on the files header signatures. Examples of different file categories are images, binaries, ASCII text, compressed, encrypted, etc. Autopsy also compares the files header signature with the file extension and will report any files that have mismatched extensions. This feature is useful if the investigator is interested in browsing files of a specific type. Graphical images will be sorted and displayed with thumbnails making it convenient for the investigator to quickly identify any images with illegal content. The TSK tool used to sort files by file type is *sorter*.

e. Keyword Searching

This powerful feature allows the investigator to search allocated or unallocated space for a specified keyword. The keyword can be searched for in ASCII or in Unicode. Autopsy also supports keyword searching using regular expressions [Ref 27]. Regular expressions, for example, are useful for finding string segments that match the format for credit card numbers, social security numbers, URLs, etc.

f. Timeline

This feature of Autopsy will generate a timeline based on the MAC times of all files on one or multiple partitions. MAC times are available for the three data types; allocated files, unallocated files, and unallocated meta data structures. For UNIX file systems the investigator can specify which volume contains the */etc/passwd* and */etc/group* files so that user and group names are displayed on the timelines instead of user and group IDs. A range of dates can be specified for creating timelines. This allows investigators to focus on data that may have changed around the time of an incident. A timeline of file activity may aid the investigator in identifying suspicious activity. Timeline files are generated through the combination of *ils*, *fls* and *mactime* found in TSK.

g. Meta Data Analysis

Autopsy allows the investigator to view the contents of allocated and unallocated meta data structures. Allocated meta data structures are useful for determining the MAC times of a file. If a meta data structure is unallocated it may also be useful for recovering deleted files. The TSK tool used in meta data analysis is *istat*. For recovering files *icat* is used.

h. Data Unit Analysis

The investigator can view the contents of a data unit in ASCII, strings, or hex format. This mode of analysis is useful for looking at fragments of files that may have been deleted or for looking at the contents of *slack space*. The investigator also has the option of examining unallocated space by itself. The TSK tool used to view the contents of a data unit it *dcat*.

i. Unallocated Space and Strings Extraction

Disk images can be quite large and overwhelming for analysis. Autopsy allows for certain types of information to be extracted from the original

disk image so that analysis can be performed on a subset of data. Unallocated space (segments of the disk that still contain data, but have been marked by the file system as available for reuse) can be extracted to a separate image file. Keyword searching and data unit analysis can be performed on the unallocated image. ASCII strings and Unicode strings can be extracted to different image files as well. The investigator can then specify which strings image to perform keyword searches on. Autopsy uses the TSK tool *dls* to extract unallocated space, the Unix tool *strings* to extract ASCII strings and the TSK tool *sstrings* to extract Unicode strings.

j. Image Details

This feature of Autopsy displays the details of the file system, such as file system type, block unit size, times of activity, and disk layout. Knowing the layout of the file system can help the investigator to identify places to look for evidence.

Together, The Sleuth Kit and Autopsy offer an excellent alternative to commercial applications TSK and Autopsy can be used to analyze any type of disk image, including some of the less documented file systems such as NTFS. TSK and Autopsy also compile and run on many different platforms including, FreeBSD, Solaris, OSX, Linux, and will now even run under Cygwin in Windows [Ref 15]. Autopsy can even be used from a CD for live analysis making it portable to the actual scene of the crime or incident.

TSK and Autopsy as a package is a leading product for digital forensic analysis. With the addition of automated case reports that thoroughly detail the analysis performed, TSK and Autopsy should appeal even more to the professional investigator.

THIS PAGE INTENTIONALLY LEFT BLANK

III EHNANCEMENTS FOR CASE MANAGEMENT AND ANALYSIS IN AUTOPSY

With new enhancements to current features in Autopsy version 2.05 more meaningful reports can be produced. There are missing features in Autopsy version 2.05 that restrict the capacity for automatic case reports to be complete. This chapter will focus on describing what was modified and extended in Autopsy with support for case reports in mind. These enhancements range from minor to significant. The version of Autopsy with the changes proposed in this chapter will be referred to as version 2.05_RHC beta, or just 2.05_RHC for short. This is not an official release from Autopsy's author. Autopsy 2.05_RHC beta, is currently only an experimental release for research purposes. The following sections describe the portions of the Autopsy user interface that were modified in this thesis.

A. CREATE A NEW CASE

The investigator will notice that the Create a New Case page as been changed in version 2.05_RHC to support better descriptions of the case and of the investigators.

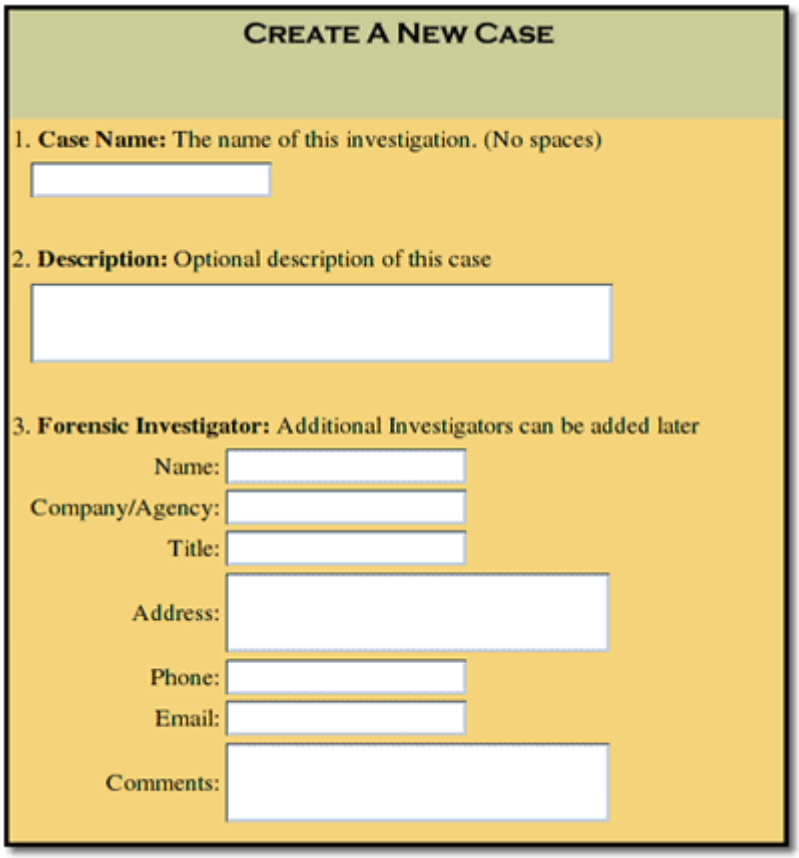


Figure 2. Create a New Case in Autopsy version 2.05_RHC beta

1. Case Description

The first minor change is that the investigator is no longer restricted to 32 characters for the case description field. The case description field will now effectively support an unlimited amount of characters. New lines in the description are now supported and will not cause an error as they did in version 2.05. The reasoning for this change is that the investigator should be free to elaborate as much as needed for describing the case. The case description may be a necessary detail of the case report. Examples of a case description may detail a case number, where the physical evidence was obtained, who submitted the evidence, and an overall synopsis of the nature of the investigation.

Descriptions for a new case had to be supported in the case configuration file named *case.aut* and on the Case Gallery page. When Autopsy's Case Manager parses the *case.aut* file it expects to find specific keywords at the beginning of each newline. This is why carriage returns in the case description field were unsupported by earlier versions of Autopsy. With version 2.05_RHC, before the value of the description is written to the *case.aut* file, all new line/carriage return pairs are replaced with the HTML
 tag. This allows the case description to be output to an HTML page in the same format that the investigator originally used.

Since case descriptions may be very long, the Case Gallery page in version 2.05_RHC will only display the first 32 characters of the description. The investigator must click on the *details* link on the Case Gallery page or generate a case report to view the full description.

2. Forensic Investigator

The major change to the Create a New Case page is with the method used to enter investigator information. Autopsy version 2.05 has ten fields for entering up to ten investigator names. One issue with this is that all the investigators have to be added at the same time that the new case is being created. A new investigator can not be added at a later time unless the *investigators.txt* file is edited manually. Also, version 2.05 does not support investigator names with spaces. A professional case report should include the full names of the investigators for completeness.

With Autopsy version 2.05_RHC all these issues have been resolved. In addition, optional information about the investigator can be entered. Investigators can now specify their company or agency, job title, work address, work phone, email, and any other comments. Detailed information regarding the investigators is important for a case report to be complete. At the time a new case is created, only one investigator with his information can be entered. Additional investigators can be added from the Host Gallery page.

The investigator should use their full name (the period character for Middle names, suffixes, etc. is supported). The original *investigators.txt* file and its implementation have been left unmodified and will contain a list of all investigator names. The investigator's personal information will be saved in the case directory to a text configuration as *investigatorName.aut*. An example of the contents "*Dick Tracy.aut*" is given below:

comp Some Agency
title Private Investigator
addr Some Address
SomeCity, NA 99999
phone (555) 555-5555
email dtracy@someagency.com
comm Specialty: Digital Forensics, Network Forensics

The words in bold font (for demonstration only) are the keywords used for parsing by Autopsy's Case Manager, the main module responsible for managing case configuration files.

Autopsy's feature to support multiple investigators in its Case Manager is unique and not implemented by all of Autopsy's commercial competitors. It is not uncommon that a team of investigators is asked to collaborate on a case. This feature allows for independent auditing of each investigator and should be included in a case report. Support for multiple investigators works well because of the client/server model that Autopsy is based on. A team of investigators can work from independent workstations (the clients) and connect to a main server that contains the image and case files

B. HOST GALLERY

Figure 3. Host Gallery in Autopsy version 2.05_RHC beta.

1. Add Host

In Autopsy version 2.05_RHC an investigator name must be selected from the drop down menu to open a host, add a new host, or create a case report. In version 2.05 an investigator did not need to be selected when adding a new Host. All activity by the investigator should be recorded so there are no holes in the activity logs of the case report. Two new elements in the host configuration file, *host.aut*, are implemented in version 2.05_RHC so that the addition of a host into evidence is better recorded. The two new elements are a creation time (the time the host was added to the case) and an investigator name.

created Sun Jul 17 22:30:24 2005
invest Dick Tracy

2. Host Description

The host description field on the “Add Host” page now supports multiple lines as described for case descriptions.

3. Add an Investigator

The new “Add Investigator” link can be followed to a web form for adding a new investigator to the case. The investigator's information will be added and stored as described in the previous section for creating the first investigator. Duplicate names are not permitted and will result in an error.

4. Case Reports

The Case Report button has been added to the Host Gallery page. At any time, an investigator may use this link to view a case report of the current state of the investigation. The case report will be described in detail in Chapter IV.

C. FILE ACTIVITY TIMELINES

From the Host Manager page the investigator has the option to view or create file activity timelines. A timeline file may be generated from any number of volumes (partitions) the investigator chooses. With the size of current hard drives the sheer volume of these timeline files can be overwhelming. It is not uncommon for a file to be several megabytes in size. For example, a timeline file generated from a 10 GB disk containing only a default install of Fedora Core 2 was 4.6 MB in size.

The HTTP protocol that Autopsy is built on does not support displaying the contents of such large files very well and results in slow page loading times. It is possible to view timelines in Autopsy, but Autopsy itself recommends against doing so, instead suggesting the use of external text viewers. Autopsy version 2.05 does offer a few options to make viewing timelines more manageable. A summary page contains a monthly report broken down by day listing each file with recorded activity in the given month. Each month listing is a hyperlink to a page where the investigator can actually look at a section of the timeline for that month alone. From within the Timeline view the investigator may also jump to

any other month and year using a series off drop down menus. While these features somewhat mitigate the file size problem there is still no real benefit in version 2.05 for the investigator to use the Timeline feature in Autopsy. Version 2.05_RHC addresses this issue and adds additional functionality for the viewing of timelines in Autopsy to better serve the investigation.

The first change in Autopsy version 2.05_RHC was to allow the investigator to specify a date range and not just an entire month when viewing a timeline. This was accomplished by simply adding start and end day pull down menus. It should be noted that there is no reason why this could not be extended to specific hour and minute values as well. It may be the case that an investigator wants to focus on an event within a specific a time frame that he is tracking from IDS or Firewall logs. Being able to specifically define a window within the context of these large timelines is required so that the investigator does not have to suffer from the long loading times of unnecessary data.

The Timeline Summary page was modified to take advantage of the ability to specify a range of dates. The individual days (not just the months) in the summary report have been formatted as hyperlinks for viewing the timeline with the start and end day for that specific date.

Even with an enhanced ability to zoom in on finer portions of a Timeline, there still needs to be greater purpose for the investigator to choose Autopsy for timeline analysis and not rely on external viewers. Autopsy version 2.05 is limited to simply dumping the contents of the timeline to the screen (Figure 4). An instance of activity on a timeline in version 2.05_RHC now provides links for meta data analysis, file analysis and for adding an investigator's note or sequencer event (Figure 5). This removes the extra steps an investigator originally had to take to manually go to the different section of Autopsy for viewing the contents of a file and creating a note. This is especially useful in the event of identifying an unallocated file on the timeline whose file name is no longer known by the file system. The investigator can now simply click the file analysis link to quickly determine the type and contents of the file. Any instances

on the timeline that are of interest to the investigator can easily be added to the investigator's case notes or sequencer events from within the timeline view itself.

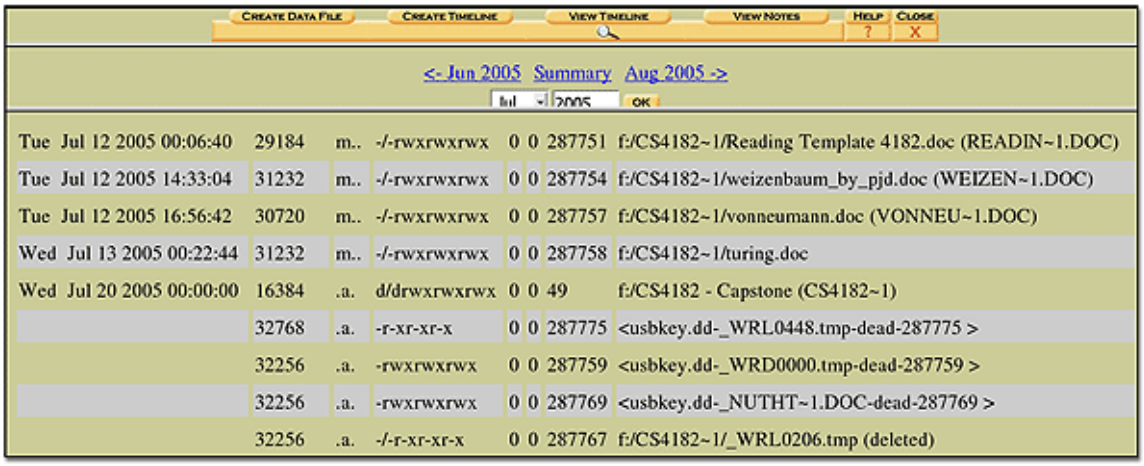


Figure 4. View Timeline in Autopsy version 2.05.

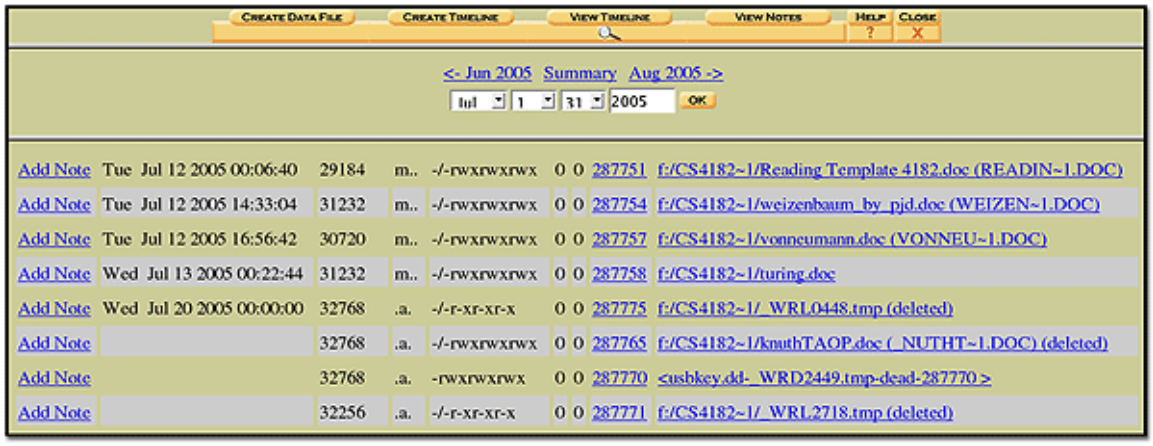


Figure 5. View Timeline in Autopsy version 2.05_RHC beta

D. KEYWORD SEARCHING

Keyword searching is a powerful feature for any forensics application. Autopsy supports searches for ASCII or Unicode strings and will accept regular expressions for either [Ref 27]. A list of keyword searches performed in an investigation and their results should be recorded in the case report.

Enhancements to the keyword searching feature in Autopsy version 2.05 were incorporated into version 2.05_RHC to make results more meaningful and better documented in case reports.

In Autopsy version 2.05, the list of search hits does not make it immediately clear in what context the keyword was found. Only the data unit number and the byte offset at which the keyword was found are reported. A link is provided for the investigator to view the contents of the data unit in either ASCII or Hex. As shown in Figure 6, the investigator must click this link and then must click yet another link to locate the meta data information to determine a filename for the file that the search result was in.. This can be a tedious process. The investigator potentially has to spend a lot of time to view the results of each keyword match to know where that match was found. Certain search hits could immediately be ruled as important or not based on the filename associated with the search hit.

The solution to this issue in version 2.05_RHC involves running the TSK tools *ifind* and *ffind* on each search hit. The meta data structure and the data unit value is determined with *ifind*. Once the meta data structure is known, the file name is determined with *ffind*. It is possible that the meta data structure cannot be determined from the data units (the contents of the meta data structure has been deleted or it has been reallocated to another file), in which case the investigator will have to use an alternative method to associate the search hit with a particular file.

If Autopsy is able to determine a meta data structure and filename then those values are used to create hyperlinks for viewing the details in Autopsy's Meta Data View and File Analysis View (Figure 7). For example, an investigator may click on the meta data structure link to verify when the object containing the keyword was last accessed. The investigator may click on the filename to view the contents of the entire file containing the keyword.

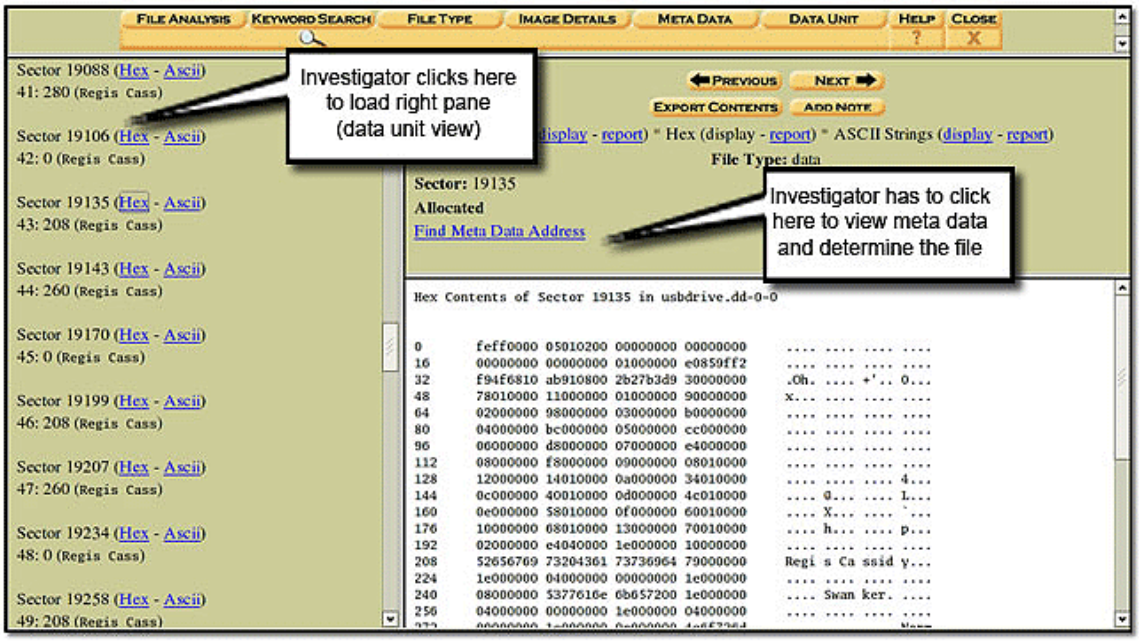


Figure 6. Keyword Search in Autopsy version 2.05.
Results are shown in left pane.

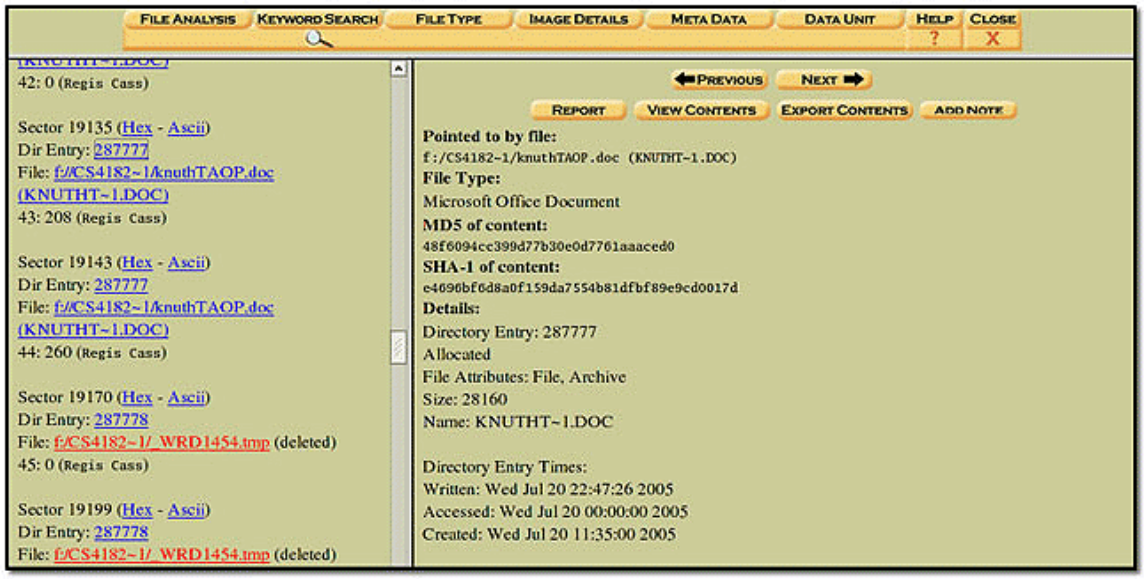


Figure 7. Keyword Search in Autopsy version 2.05_RHC beta.
Results in left pane include links to Meta Data View and File Analysis View

With the program taking extra steps to automatically determine meta data and filename information there will be more overhead involved with the keyword searching process. However, the investigator is saved a lot of time by not having to go through a series of clicks for each search result to determine if the file is of

interest or not. The investigator may still have to carefully analyze the contents of a file before determining if it is evidence or not, but most likely a good number of files can be flagged or ignored based solely on their filename.

E. FILE TYPES

Files on the disk image can be sorted based on file type. This can be useful to the investigator for singling out specific types of files that are of interest such as encrypted or graphic files. This feature of Autopsy can only be run if a file system has been determined and is readable. This feature should not be confused with various disk carving techniques [Ref 28, 29] which attempt to extract files from a raw image based on their signatures, even if a file system does not exist.

Behind the scenes of Autopsy's interface, the TSK tool *sorter* is used to categorize files. In Autopsy version 2.05 there is really little benefit from running the *sorter* utility from within the application other than the fact that the GUI eliminates the need for the investigator to familiar with the *sorter* command line. This feature in version 2.05 is seriously limited because the investigator is unable to view the results from within Autopsy. The results are stored as a series of HTML files and the investigator must view these pages outside of Autopsy. Since the analysis of these sorted files is not managed by Autopsy's Case Manager there is no way to integrate them with other Autopsy features such as meta data and file content viewing and the creation of notes and sequencer events.

File Type sorting is new and powerful feature of Autopsy version 2.05_RHC. The investigator is now able to view the results from sorting files from within Autopsy itself. The index of the Results page shows the number of files for each file category.

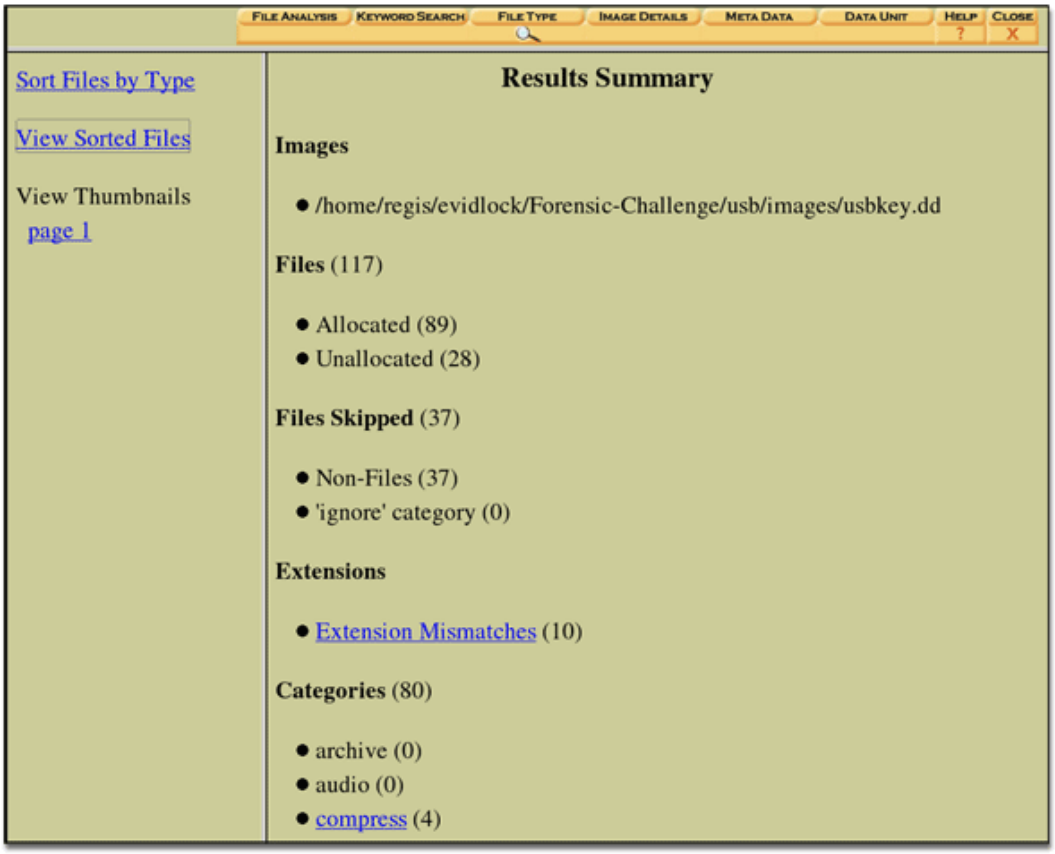


Figure 8. File Type sorting in Autopsy version 2.05_RHC beta

Examples of file categories are archive, audio, and compressed files. File extension mismatches are also reported. The number associated with each file category is also a hyperlink to a page detailing the results of the matches for that file type. Details include the filename, file type information reported from the tool *file*, the name of the disk image, meta data structure value, and the ability to create a note or sequencer event.

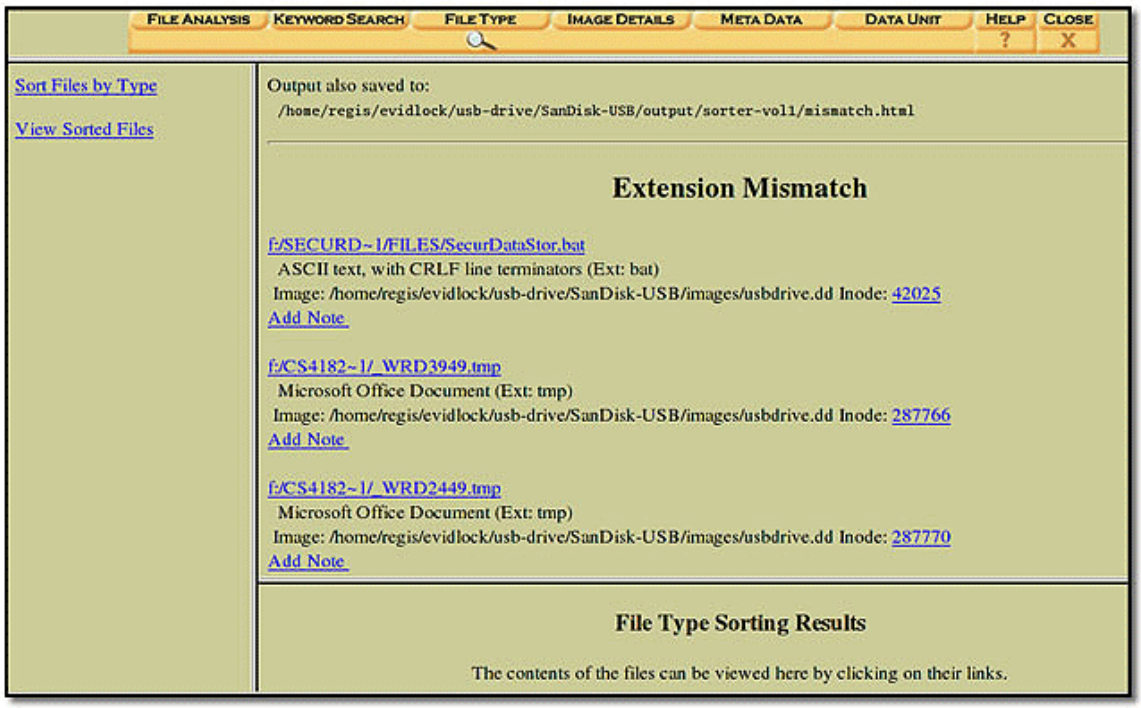


Figure 9. Viewing the results of a specific category after file type sorting in Autopsy version 2.05_RHC beta

When sorting files by file type the investigator has the option to sort only graphic image files. Copies of any graphic images found are saved to the data directory associated with the case and HTML pages for viewing the thumbnails are generated. This feature is available in version 2.05, but again the investigator is unable to view the results from Autopsy. With version 2.05_RHC the investigator now has the option to view the thumbnails for any graphic files found. The investigator can quickly identify any images of interest, such as those containing illegal content. The images themselves and their file names serve as hyperlinks to the page containing the full-sized image. The investigator can quickly add a note or sequencer event from the thumbnails page.

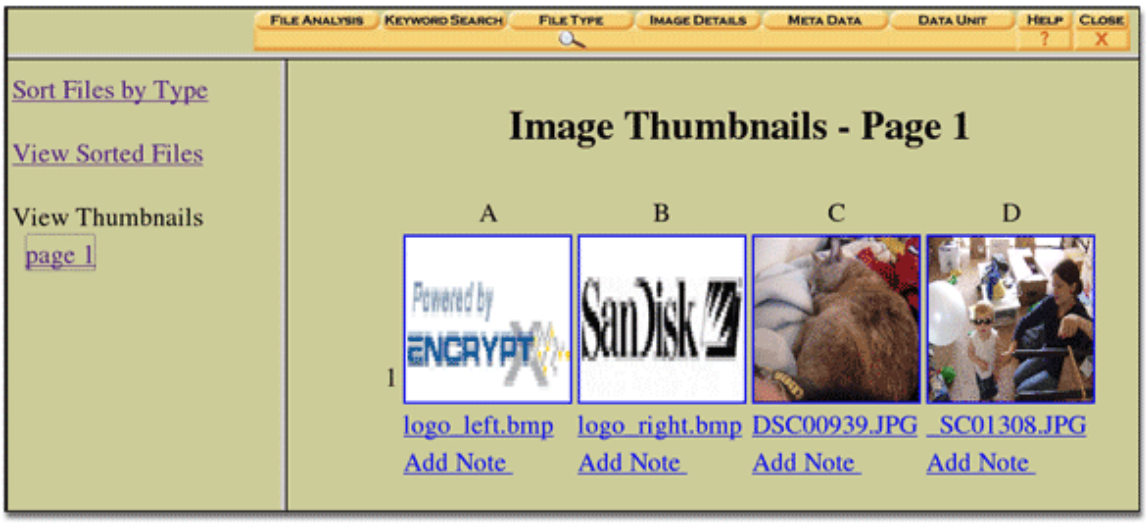


Figure 10. Sorting graphical image files in Autopsy version 2.05_RHC beta

Autopsy version 2.05_RHC parses the HTML files created by *sorter* for displaying for viewing the sorted files and thumbnails. These files are located in the data directory of the case. Linking directly to *sorter's* HTML files is considered insufficient because there would be no capability for jumping directly to the file analysis and meta data modes for those files. The ability to directly create an investigator's note would also not be available.

F. BOOKMARKS, NOTES AND SEQUENCER EVENTS

To allow for better organization during the investigation process a significant addition to the way notes and sequencer events are managed is implemented in Autopsy version 2.05_RHC. The investigator can now associate a bookmark name with each note or sequencer event. Bookmarks are used to categorize different pieces of evidence into related groups. For example, the investigator may create a bookmark named "Rootkit." Throughout the investigation the investigator may find evidence such as an ftp client being accessed (to download the rootkit), a directory of *untarred* files, various trojaned system tools, etc. As the investigator adds notes and sequencer events for these

items of evidence they could be bookmarked under “Rootkit”. Chapter IV E.2 provides suggestions for grouping evidence with bookmarks in a systematic way to build a partial model of the investigation. It will also be seen in Chapter IV that the case report can be used to reconstruct a sequence of events based on a bookmark category.

When adding a note and sequencer event, the investigator now has a drop down list of existing bookmarks to choose from or the investigator can use the text field to create a new bookmark. A bookmark name does not have to be specified and any notes created without one will be grouped together in the case report.

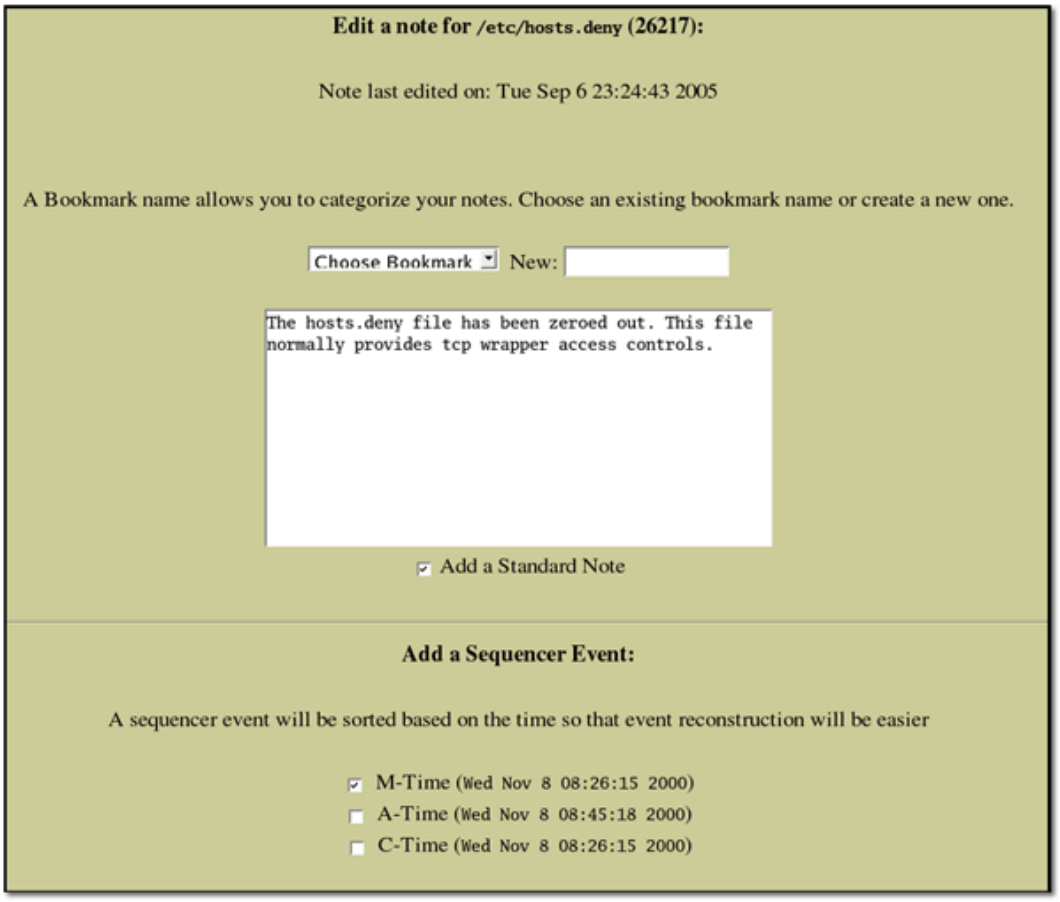
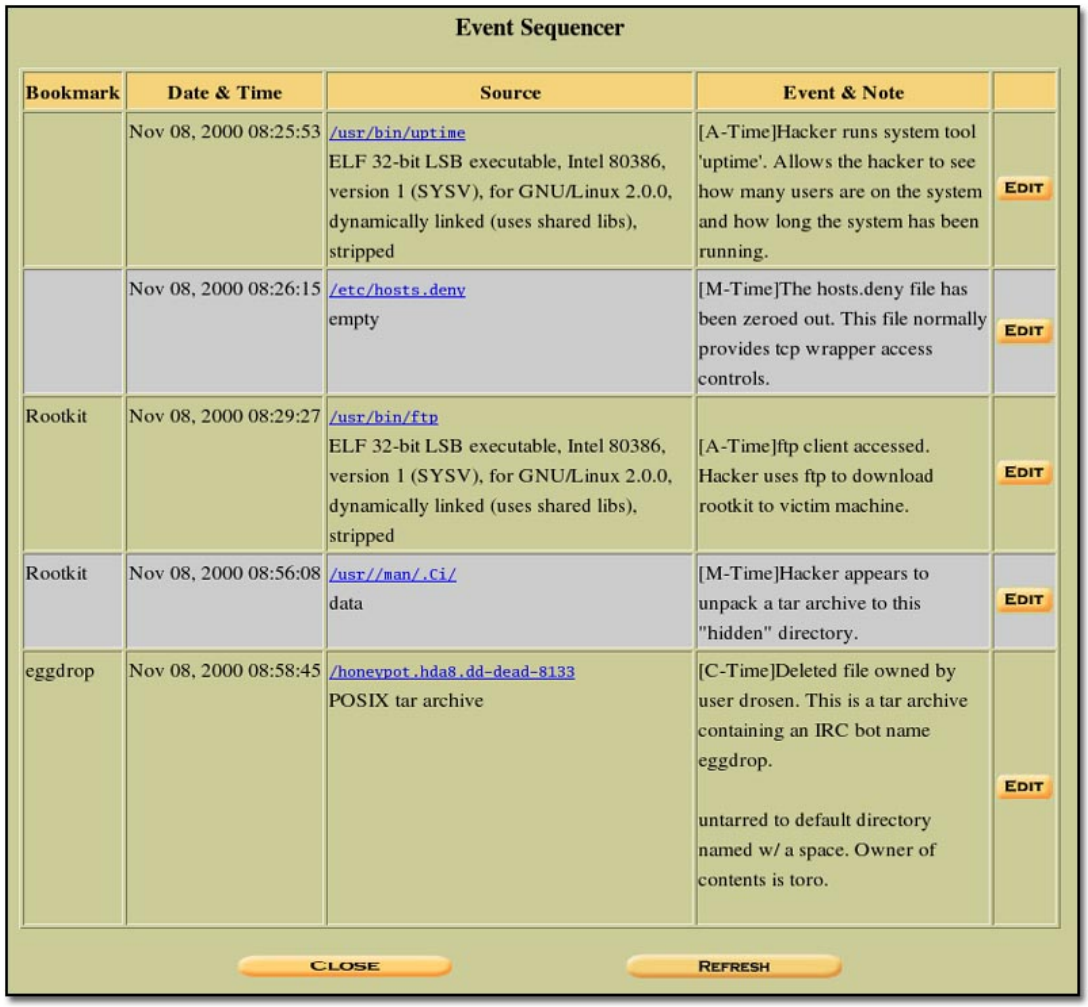


Figure 11. Creating or editing a note in Autopsy version 2.05_RHC beta

In Autopsy version 2.05_RHC, if a note or sequencer event is created for an actual file, then the output from the Unix utility *file* will be saved as well. The

file utility is used to determine the type of file such as a Microsoft Word document, a jpeg image, etc. This is not considered to be computationally expensive, since notes can only be made one at a time and the output of *file* is usually computed in a fraction of a second. Including the known file type for bookmarked evidence should be considered important for the case report.



Event Sequencer

Bookmark	Date & Time	Source	Event & Note	
	Nov 08, 2000 08:25:53	/usr/bin/uptime ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.0.0, dynamically linked (uses shared libs), stripped	[A-Time]Hacker runs system tool 'uptime'. Allows the hacker to see how many users are on the system and how long the system has been running.	EDIT
	Nov 08, 2000 08:26:15	/etc/hosts.deny empty	[M-Time]The hosts.deny file has been zeroed out. This file normally provides tcp wrapper access controls.	EDIT
Rootkit	Nov 08, 2000 08:29:27	/usr/bin/ftp ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.0.0, dynamically linked (uses shared libs), stripped	[A-Time]ftp client accessed. Hacker uses ftp to download rootkit to victim machine.	EDIT
Rootkit	Nov 08, 2000 08:56:08	/usr//man/.Ci/ data	[M-Time]Hacker appears to unpack a tar archive to this "hidden" directory.	EDIT
eggdrop	Nov 08, 2000 08:58:45	/honeypot.hda8.dd-dead-8133 POSIX tar archive	[C-Time]Deleted file owned by user drosen. This is a tar archive containing an IRC bot name eggdrop. untarred to default directory named w/ a space. Owner of contents is toro.	EDIT

CLOSE REFRESH

Figure 12. Viewing sequencer events in Autopsy version 2.05_RHC beta

Autopsy version 2.05 lacks the ability to modify a note or sequencer event after it has been created unless the investigator modifies the plain text file used to store notes manually. As a matter of usability, the underlying configuration files should be transparent to the investigator. Therefore, the investigator needs to be able to edit notes and sequencer events from within Autopsy. Throughout the

course of the investigation it is likely that an investigator may continue to learn more about a specific piece of evidence and will need to make additional notes. Being able to edit a note also allows the investigator to assign bookmark names at a later time when more information is learned.

Autopsy version 2.05_RHC beta has an edit button on the Notes page and on the Event Sequencer page next to each entry. When the investigator edits a note or event he is taken to the same kind of web form that was used to create the note in the first place, but with fields filled with the existing information. The investigator may change any of the values he wishes. Any existing entries in the notes or event sequencer will be removed and replaced with the new information.

To accommodate the changes in the way notes and sequencer events are created, the file type and bookmark name are appended to the existing entry lines in the *.notes* and *.seq.notes* files. These files are found in the data directory of the host directory and are pre-appended with the investigator's names. All new bookmark names created are stored to *bookmarks.txt* found in the case root directory.

The modifications described from Autopsy version 2.05 to version 2.05_RHC are considered essential for creating well structured and complete case reports. Much focus has been placed on the usability for creating notes and sequencer events to mark evidence. With the enhanced version of Autopsy the investigator can now create a note from all analysis modes that apply. The usage and organization of notes, sequencer events, and now bookmarks will determine the usefulness of the case report. The next chapter describes the features of case reports now supported in Autopsy version 2.05_RHC beta.

THIS PAGE INTENTIONALLY LEFT BLANK

IV THE CASE REPORT

The primary addition to Autopsy in version 2.05_RHC beta is the ability to generate case reports. The case report will provide an overview of the digital forensics and analysis performed with Autopsy. The Autopsy program is composed of several different modules written in Perl. Separate modules have already been written for each of the different functions in Autopsy; functions such as file sorting, keyword searching, timeline analysis and case management. A new module, named *Reports.pm* (Appendix B), has been added to Autopsy to support the function for creating case reports.

One of the motivating factors for creating case reports is to provide a single source of documentation that details all of the different components of a given case. Recall that case management in Autopsy supports the addition of multiple hosts and multiple disk images within a single case. The analysis of those disk images may be conducted by one or more investigators. Because the analysis can be spread across so many different components of the case, interpretation of the results can be disorganized and difficult to follow.

Results from the various forms of analysis in Autopsy will be merged so that the investigation can be reviewed from a holistic point of view. Prior to the support for case reports, each investigator was limited to a view of his own activities for a single host. For example, in Autopsy version 2.05, a user is only able to view the notes, sequencer events, and keyword searches that belong to the investigator that he logged in as (the investigator chosen from the pull-down menu on the Host Gallery page). If the case involves multiple hosts, the user may then only review those results one host at a time. Running multiple instances of Autopsy or viewing actual print outs side-by-side is the only way to review the case in its entirety and even then the results are not visually ordered in a way that would help make sense of the data. Because the case report provides a view to the mergence of all host and investigator information, it also

serves as an important analysis tool. Once the investigation is complete the case report can then be handed over to any party that is interested in its review.

This chapter details the different sections of the case report that can now be generated in Autopsy version 2.05_RHC beta. The report is generated as a set of web pages using HTML, much like the case reports that can be generated by FTK version 1.5. A web page gives the user the ability to easily browse the various sections of the report. Navigation through the case report is made possible via hyperlinks found in the case menu.

The case menu is divided into three parts described in the following sections. A few select figures that illustrate example pages from a case report are shown in this chapter. For a detailed example of a complete case report the reader should refer to Appendix A

A. CASE INFORMATION

The details regarding different components in the case can be found in the Case Information section of the case report. Components include the investigators, hosts, disk images, disk partitions, and special volumes that have been extracted. Examples of volumes extracted from a disk image are timeline, string, and unallocated files.

1. Case Map

The Case Map (Figure 13) is a diagram of all the host, image and partition components of the case. It resembles a hierarchal tree structure and illustrates how the components relate to each other. An investigator, or anyone else reviewing the case, can refer to the Case Map to get a quick visual representation of what has been added into evidence. The Case Map is also the quickest way to reference the details about a particular host or volume.

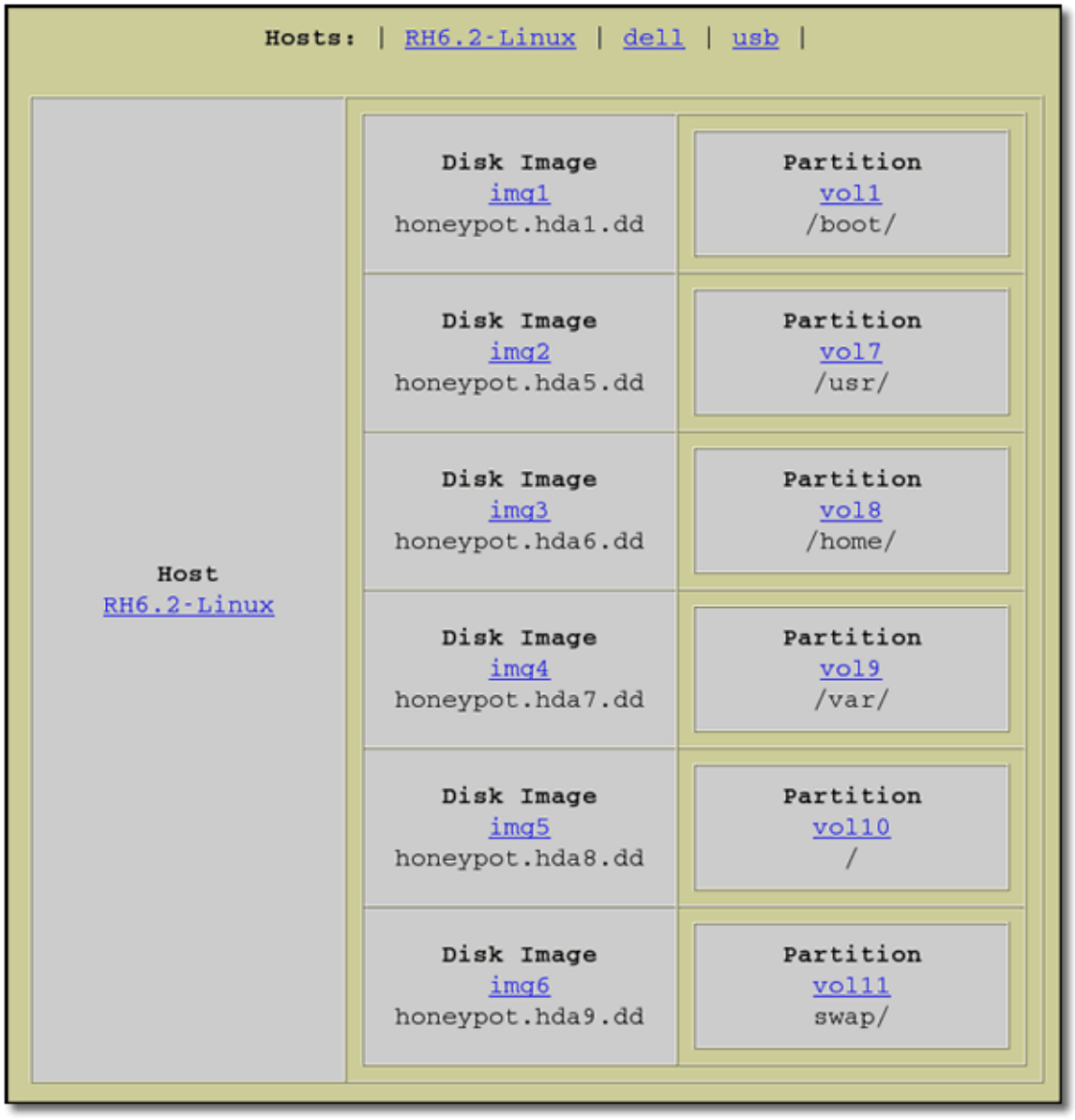


Figure 13. Example of a Case Map in the Autopsy case report

2. Case Details

The details of the case can be viewed by clicking the Case Details link on the case menu. The format for the Case Details page is shown in Figure 14.

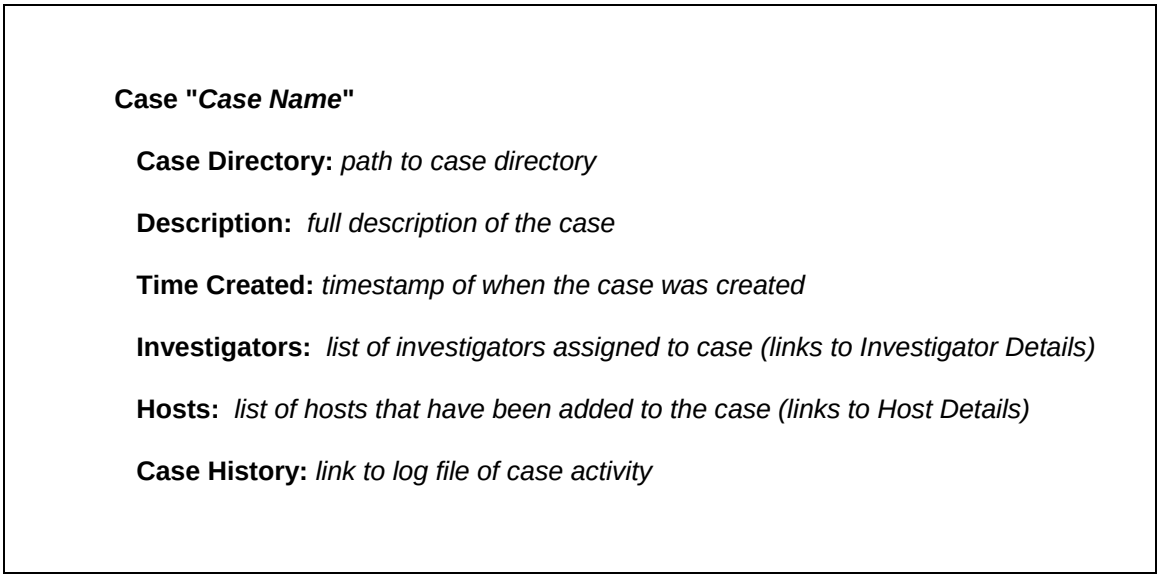


Figure 14. Format for the Case Details page of the Autopsy case report

The source for the case description field is produced by the investigator at the time the case is initially created. The investigator should make use of the case description to describe the pertinent parts of the case that should be included in the case report.

3. Host Details

The details for a specific host can be viewed by following the link to the host from the Case Map or from the Case Details page. The format for the Host Details page is shown in Figure 15.

Host "*Host Name*"

Host Directory: *path to host directory*

Description: *full description of the host*

Time Added: *timestamp of when the host was added to the case*

Investigator: *Investigator that added the host to the case (links to Investigator Details)*

Host Timezone: *timezone the host was in when confiscated*

Timeskew: *number of seconds the host clock was off from real time*

Alert Hash Database: *path to file containing hashes for known bad files*

Exclude Hash Database: *path to file with hashes for files to be ignored in analysis*

Images: *list of disk image files added to the host (links to Image Details)*

"Img ID" (i.e. img1) "file name" (i.e. image.dd)

Timeline Files: *table of timeline files and their details*

Example:

File Name	Type	Created	Investigator
body	*Body*	*Fri Jul 29 23:26:42 2005*	*Regis H. Cassidy*
timeline.txt	*Timeline*	*Fri Jul 29 16:27:24 2005*	*Regis H. Cassidy*

Case History: *link to log file of host activity*

Figure 15. Format for the Host Details page of the Autopsy case report.

4. Image Details

The details for a disk image can be viewed by following the link from the Case Map or from the Host Details page. The format for the Image Details page is shown in Figure 16.

Image "*Disk Image Name*" on Host "*Host Name*"

Image Path: *path to the disk image file*

Time Added: *timestamp of when the disk image was added to the host*

Investigator: *Investigator that added the host to the case (links to Investigator Details)*

Volume ID: *the volume ID number assigned by Autopsy*

MD5: *MD5 hash value of image file name*

Partitions: *list of all partitions on the disk images (links to Partition Details)*

"Vol ID" (i.e. vol1) "mount point" (i.e. c:) "file system" (i.e. NTFS)

Figure 16. Format for the Image Details page in the Autopsy report

5. Partition Details

The details for a partition can be viewed by following the link from the Case Map or from the Image Details page. The format for the Partition Details is shown in Figure 17.

Partition "*Partition Name*" on Image "Image name"

Mount Name: *name of the mount point specified when adding the disk image*

File System: type of file system, such as ntfs, fat, ext ,ect. (links to file system details)

Volume ID: *the volume ID number assigned by Autopsy*

Sectors: *the beginning sector and last sector that mark the bounds of the partition*

Extracted Files

Tables describing the various volumes that may have been extracted from the partition.

Example:

ASCII Strings			
Vol ID	**Name**	**Created**	**Investigator**
vol9	*Image.dd-0-0-ext.asc*	*Thu Aug 25 22:09:04 2005*	*Regis H. Cassidy*
		MD5: A2F1F4170BD860B54949D293D8EB20FE	

Figure 17. Format for the Partition Details page for the Autopsy case report

6. File System Details

Every partition either contains a file system or it is considered to be a raw volume. The details for a file system can be viewed by following the file system details link from the Partition Details page. The information regarding the file system is collected from the output of the TSK tool *fsstat*.

7. Investigator Details

The details for the individual investigators can be viewed from the Investigator Details link on the case menu. The information that was recorded when the investigator was first added to the case is shown in this part of the report. This includes the investigator's name, agency, job title, work address, phone, email and any additional comments. Also included in the Investigator Details page are hyperlinks to the log files generated automatically for each investigator and the notes they took throughout the course of the investigation. There are two log files kept for each investigator. One is a history of the actions performed in Autopsy and the other is a complete list of TSK commands that were invoked by the Autopsy interface. These logs can be used to retrace the steps of an investigator's analysis if needed.

B. CASE AUDIT

The Case Audit section is used to verify the integrity of the investigator's analysis. Proper log files must be recorded and acquisition hash values for the disk volumes must be computed.

1. Log Files

Under the Case Audit section on the menu there is a link to a page containing all the log files recorded in the investigation. This page is a source to the individual logs files that have been generated for the case, each host, and each investigator. The log files are grouped by host name. Utilizing these log files a detailed history of the analysis performed with Autopsy can be pieced back together, if needed. The Case Details, Host Details, and Investigator Details pages also provide links to their respected log files.

2. Image Integrity

To maintain credibility in a court of law, an investigator must demonstrate that the disk images subject to his analysis were absolutely unmodified. One way of doing this is to compute a unique hash value of a disk at the time of acquisition. Autopsy can be used to compute these hash values.

The Image Integrity link on the case menu is a page that provides the MD5 hash values of any disk image or volume that was computed by Autopsy. MD5 values may be available for any disk images added to the case, unallocated space images, ASCII string files (from both allocated and unallocated images), Unicode string files (from both allocated and unallocated images), data timeline files, and the actual human readable timeline files. The list of images and volumes and their MD5 values are grouped by host.

C. ANALYSIS INFORMATION

The analysis information section of the case report contains the results from the analysis in Autopsy. The content of these pages may be useful to the investigator even though he has already seen the output from running his analysis in Autopsy. In this section of the case report an investigator can view the results of his examination in the context of the other results from other hosts and from other investigators.

1. Case Notes

Each investigator in Autopsy can create a note for a file, meta data structure, or series of data units. The Case Notes page contains all the notes recorded by every investigator. The note entries are grouped together by their bookmark name. Bookmark names are used so that a reviewer can easily identify pieces of evidence that are related to each other. The case report

feature in Autopsy version 2.05_RHC is the only way to view groups of notes that are merged together from multiple hosts and created by different investigators.

Navigation to a particular bookmark group is made easy with a bookmark navigation menu on the Case Notes page. The “default” category is a link to those notes for which no bookmark was specified. Each note entry includes the investigator and host that the note was created from. The investigator name serves as a hyperlink to view only the notes created by that particular investigator.

2. Sequencer Events

The Sequencer Events page is similar to the Case Notes page except that the entries are ordered by MAC times. The investigator has the option, when creating a note, to also create a sequencer event for any of the three MAC times. This page serves as a timeline of file activity for all the files that have been marked as evidence.

If the “All Events” link is chosen from the bookmark navigation menu then all the events created for the case will be merged into a single timeline. Alternatively, a specific bookmark category can be chosen from the bookmark menu. It may be useful to generate timelines for specific groups of related events. This is possible only if the files or events are bookmarked under the same category.

3. Keyword Searches

The Keyword Searches page of the Case Report will show all keyword searches that were performed in the case. A table is constructed so that the keywords are alphabetized. The table includes the number of hits matching the keyword, whether it was an ASCII or Unicode search, and the name of the volume and host that was searched. The keyword term in the table can be clicked to review the results of that particular search.

4. File Categories

The File Categories page of the case report contains a table showing the results from running *sorter* for any of the disk images on any of the hosts. Recall, that the investigator uses the File Types section of Autopsy to run the *sorter* utility. The rows in the table represent the different file type categories and the columns identify the different hosts and volumes that belong to the case. The columns are labeled with the name of the mount point of a given volume (i.e. /home, C:\, etc.). The contents of the individual cells indicate the number of files of a given type on that volume. The number value is hyperlinked a page that shows all the files belonging to that file category and their associated details. A reviewer can use this part of the report to see what files are text documents, zip archives, graphic images, etc. This page also reports any files whose extension does not actually match the actual content of the file.

5. Image Thumbnails

The Image Thumbnails page of the Case Report shows thumbnails and their respective file names for any images that were bookmarked with a sequencer event during the investigation. The thumbnails are ordered by the MAC times that were used for the sequencer event. This page essentially provides a timeline of image activity that is useful for child pornography investigations and cases of that nature. An image thumbnail may appear more than once in the timeline depending on the number of MAC times chosen when creating the sequencer event. For example, a thumbnail may be located in the timeline with its C-time or M-time indicating when the image was created or downloaded. It may appear later in the timeline with its A-time indicating when the image was last accessed or viewed. The thumbnails provide a link to a page containing the full-sized image and more of its details. These details include the type of image file, all three of its MAC times, and the note that was left by the investigator.

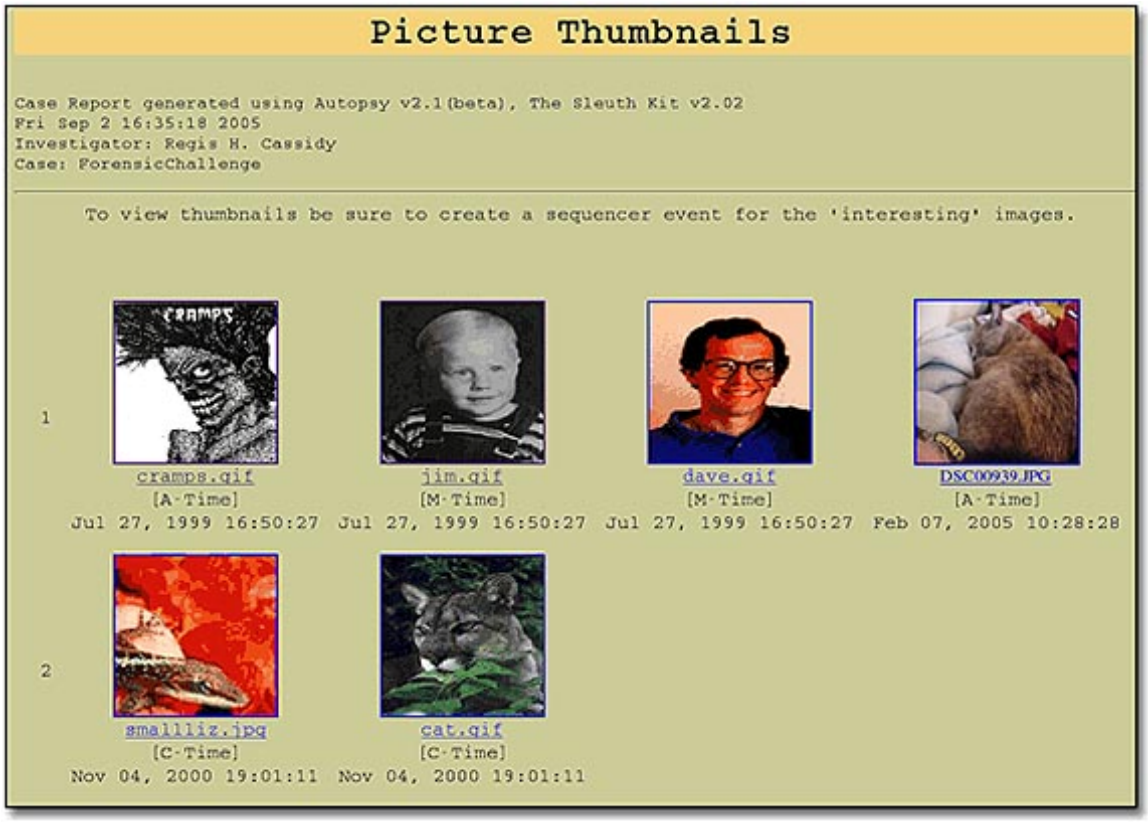


Figure 18. Example of a timeline of picture files in the Autopsy case report

6. Individual Reports

Individual reports can be created from the File Analysis or Data Unit view in Autopsy. Review Chapter II Section 1.d for an explanation of the types of individual reports that can be exported in Autopsy. If these reports are saved to the reports directory located in the host directory, as intended, then they will be viewable in the Individual Reports section of the case report. In fact, the investigator may put any type of text-based report or log in the reports directory and their contents will be included in the case report. Examples of other types of individual reports may be logs from an IDS or firewall, the output of various tools used to collect volatile information, and the output from other forensics tools such as *Foremost* [Ref 28, 29].

D. VIEWING AND SAVING THE CASE REPORT

A case report is generated when the user clicks the Case Report button from the Host Gallery page. The case report is viewed "live" in the user's web browser; that is the case report module sends data dynamically to the Autopsy server program. An investigator may have two browser windows open; one to display the case report and the other to perform analysis. As the investigator executes various actions in Autopsy the case report window can be refreshed to reflect the changes that have been made. Autopsy must be running to view the case report in "live" mode. This "live" mode allows the investigator to review the current state of the case report at any time during the investigation without exporting it to disk.

Case reports can also be exported and saved to HTML files so that they can be reviewed without running Autopsy. The investigator assigns a name for the report on the case menu (the default name is the case name). If the investigator clicks the Save Report button, all necessary files are exported to the directory *casereports* which can be found in the case directory. If the user saves the case report again with the same name, then the old report files will be overwritten. These saved reports can be archived for future review.

E. ORGANIZING YOUR ANALYSIS FOR THE CASE REPORT

In this section, we describe techniques and methods recommended for creating well organized and meaningful reports. If the investigator takes the right care while conducting his analysis in Autopsy the case reports will better express the nature of the investigation. The case report should be well organized so that the reconstruction of the events of an investigation is as effortless as possible. This is especially critical if the report is to serve as a "lessons learned" for future investigations.

1. Adding Hosts to the Investigation

Often, evidence for a computer crime may involve more then one computer or other type of digital media device. The idea behind being able to add multiple hosts to an Autopsy case is to better support an investigation where it is likely multiple computers on a network have been compromised. However, there may be other reasons for adding multiple hosts to a case other then in a network scenario. Therefore, the term “host” should be interpreted loosely and not just to mean a host on a network of computers.

One particular individual may use multiple computing devices, such as a desktop computer for home and a laptop or a PDA for working remotely. A crime may involve more then one suspect, each owning a computer that must be investigated. Evidence may be recovered from a wide range of media devices. For example, removable hard drives, thumbdrives, CDs, DVDs, digital cameras, printers, and faxes are all forms of digital media that the investigator might have to treat as evidence and analysis carefully. Each of these sources of digital media can be imaged and added to an Autopsy case to undergo analysis. All of the image files collected must be added using the Host Gallery page in Autopsy, meaning every image file has to be associated with some host.

It is up to the investigator to decide on how to group items of evidence into hosts. In the event of handling multiple hard disks installed in one computer, common sense says to create a host in Autopsy that represents the machine from which the hard disks were taken from. This could be the host name of the machine if it was on a network or some other descriptive name based on the computer model or installed OS. But what about the other media that has been collected that is not a physical component of the actual computer? CDs and other removable media can be added to the host representing the computer or a new host can be defined for them to be associated with. How the investigator decides to group items of evidence into hosts will affect the organization of the case report. Investigators should choose a descriptive host name and way to organize the image files for that host, that make the case report as clear and easy to

review as possible. This is also where the host description field comes into play and is important.

An example (Figure 19) is given below that provides an example of how an investigator may choose to organize all the collected media in his investigation and how it will be displayed in the Case Map of the Autopsy case report. This example case is an investigation of the pornographic solicitation of minors which may involve a wealth of digital media types.

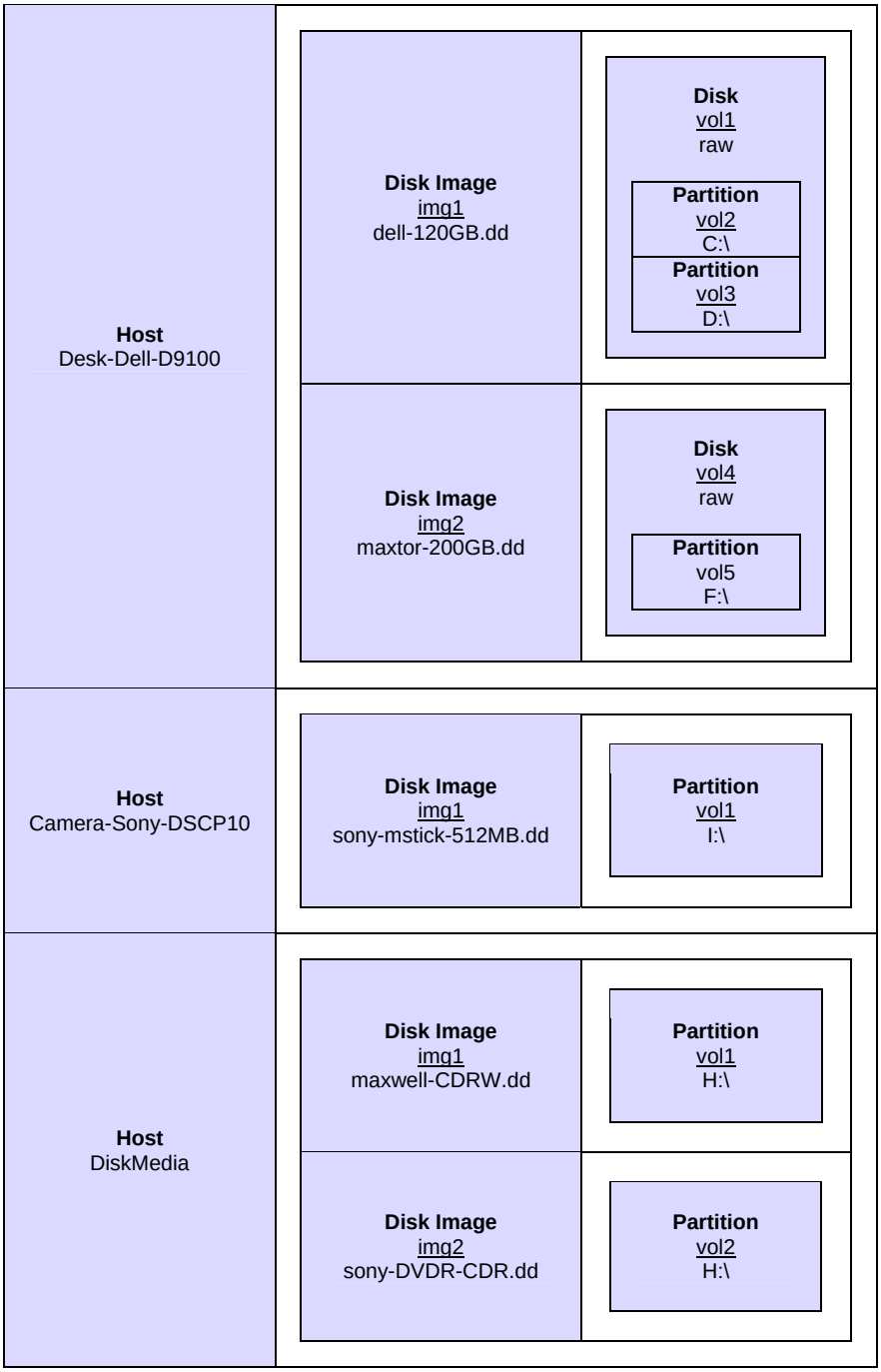


Figure 19. Case Map involving multiple types of digital media

2. Grouping Evidence with Bookmarks

Chapter III introduced the interface that showed how Autopsy version 2.05_RHC beta connects investigator notes or sequencer events with a

bookmark name. Bookmark names can significantly help in organizing data that has been identified as evidence. A bookmark name may simply be descriptive such as, “Email”, “Illegal Images”, or “Encrypted Files.” Utilizing bookmarks in this way directly relates the name to the data.

A bookmark may also represent the evidence in a more abstract way. The “Rootkit” example is a good way to illustrate this. The access time of an ftp binary and a trojaned *ls* program that hides a series of malicious files from the user do not seem to be related when reviewed independently out of context. A bookmark puts these items of evidence into context. The bookmark “Rootkit” is used to categorize all the evidence found that supports the theory that a rootkit has been installed on the system. The access time of the ftp binary indicates the hacker used the ftp protocol to download the rootkit installation files from a remote server. The trojaned *ls* binary helps to provide proof that the rootkit was actually installed on the system.

Investigators should intelligently use bookmarks to describe the evidence they collect. It is very likely that an investigator will have to manage a huge amount of information requiring them to investigate many paths. To solve a crime, one may break it up into several hypotheses [Ref 30]. The hypotheses that define criminal activity and the collected evidence that verify (or falsify) those hypotheses can be represented with bookmarks.

Ultimately, the investigator will be able to use the features of bookmarks to create a formal model of the investigatory process. The goal of the formal model is to organize forensic knowledge in a reusable way so that past experience may be used to train new personnel, share knowledge within the forensic community, and expose collected information to quality assessment by third parties [Ref 30]. More research and functionality is required for the bookmark and case report features of Autopsy before a formal model can be produced, but the work in this thesis is a step in the right direction. The future work that is needed in this area is briefly discussed in Chapter V.

THIS PAGE INTENTIONALLY LEFT BLANK

V CONCLUSIONS

A. SUMMARY

The purpose of this thesis involved the study of automating the generation of case reports for applications used in digital forensics. A successful implementation of case report generation has been added to the Autopsy Forensic Browser, creating a more robust tool for forensic examiners. An Autopsy case report allows the digital investigation to be reviewed from a higher level that encompasses the documentation, logs, and analysis results for all digital media and for all investigators.

The case report feature was engineered to support both the analysis (reconstruction) and presentation phases of any of the forensic process models. An Autopsy case report can be actively open and its content updated dynamically in conjunction with the investigator's analysis. This provides a snapshot of the progress of the investigation at any time and may assist the investigator in his understanding of the evidence. Upon completion of the investigation, the case report can be exported and distributed for review as needed.

During the design process of the Autopsy case report module it became apparent that additional components and features were needed in Autopsy's case management. Various features of Autopsy were enhanced to provide better case organization and documentation. With these modifications higher quality case reports are possible. The most significant change to Autopsy's case management is in the way the investigator can now bookmark evidence. Data found in the analysis can be labeled with a bookmark name when a note or sequencer event is created for that data. This allows various items of evidence to be categorized into logical groups based on their relationships to each other.

The work done in this thesis has served to improve the functionality and capabilities of the Autopsy Forensic Browser. Autopsy is an open source tool

which made this thesis possible and allows for continued development. Autopsy's new support for case reports behaves much like the case reports supported in commercial applications like EnCase and FTK. Autopsy is now equipped to better compete with these commercial applications.

B. FUTURE WORK

At the time of this thesis' publication Autopsy version 2.05_RHC beta has not yet been made publicly available. Following coordination with the original author of Autopsy, Brian Carrier, an official may take place in the near future. The changes made to the program with the work of this thesis need to be released and tested in the forensics community. Feedback from forensic examiners that have a need for case reports would prove invaluable for further development of the case report features in Autopsy. Expert witnesses need to be interviewed regarding the different ways they present digital evidence in court and to corporate management. Whatever the digital evidence may be, there is some translation process from its digital representation to a physical presentation in order to present the findings to a jury or manager. It is this translation process that should be automated as much as possible.

Features in Autopsy's case management and in the case report module can be further improved to augment usefulness and readability of the case report. The following sections describe some ideas that have not been implemented yet.

1. Autopsy Case Management

Ideally, one application could be used for all data acquisition, extraction, and analysis of digital media. If one application is able to successfully perform all tasks for an investigation of digital evidence then the automation for case reports is much more feasible and will be more comprehensive. For example, Autopsy

does not currently have any support for disk image acquisition. Images of the physical disks must be created with external tools like *dd* [Ref 31] and *dcfldd* [Ref 32]. Autopsy has no knowledge of what tools are used for creating disk images and therefore cannot automatically include this information in the case report. It would actually be fairly easy to have Autopsy interface with a disk imaging tool like *dd*, which is a tool installed by default on most Unix systems, to create disk images.

Other features that are not part of Autopsy, but considered to be standard practice in an examiner's investigation are disk carving, webpage and email analysis, and Windows registry analysis. If Autopsy had such functionality then the results of the analysis could automatically be included in the case report. In reality, examiners will most likely always have to use a variety of tools in their analysis, but the more that can be done with one tool, the easier it is to collectively document and manage the evidence.

The individual reports that can be created in Autopsy for files and segments of disk units are considered very useful. A more systematic way of managing these technical reports and including them in the main case report is needed. Rather than relying on the investigator to save these reports to a specific directory, they should be automatically saved to a default directory so that the case report module understands where to find them.

It is probably safe to assume that if the investigator creates a note or sequencer event for a file or data unit segment, that there is a good chance he also wants to generate a technical report for that piece of evidence. Therefore, it should be easy to implement a checkbox option on the note and sequencer event pages to also export a technical report. The investigator should be able to export these reports at the time a note is created or for a group of evidence items when viewing the pages with all the notes and sequencer events. If the reports were implemented this way then they could also be associated with a bookmark name. The case report module would then be able to group the individual reports for viewing based on bookmark categories.

2. Case Reports

As mentioned earlier, case reports in Autopsy can only be automated for analysis or notes that are performed and created with Autopsy. Before a case report can be handed out for review, additional sections may need to be included whose sources come from other forensic tools or for recording miscellaneous thoughts and conclusions of the investigator. For example, a section of the case report may need to highlight information about the running processes on the system before it was shutdown and confiscated. Also, it is likely that the investigator will want to add a conclusions statement to the case report that provides an overview of the findings of the entire case.

Investigators most likely will not want to rely on using a web form in Autopsy for manually writing a conclusions section and other reports, due to a lack of formatting control, although it can certainly be made an option. Investigators will most likely prefer to use other third party word processor applications for better control, spell checking, and conventional formatting. For this reason, it may not always be convenient to limit investigators to an HTML based case report. Future work on the case report module of Autopsy is to program the ability to export the Autopsy case report to a plain or rich text format that can be imported to a case report that is being drafted in a conventional word processing application.

Currently, generating a case report in Autopsy is all or nothing. More flexibility can be added so that the case report is customizable. Before exporting the case report the investigator could be presented with a form with checkboxes and the names of the different sections of the report. The investigator could select only those sections of the reports he is interested in exporting.

3. Representing the Investigation with a Formal Model

A computer crime is typically not unique and will often follow some common pattern. Computer crimes with common patterns can be solved with

common analysis techniques. However, investigators lack a good method for sharing the knowledge of an investigation in a reusable way.

Italian researcher, Lorenzo Martignoni and his colleagues have recently written a paper in which they describe their goal: to provide a methodology for archiving, retrieving, and reasoning about forensic knowledge. They propose a model that will assist in training new personnel, sharing knowledge among forensic communities, and providing a quality assessment of the investigation to third party reviewers [Ref 30]. Their model is based on being able to express clear hypotheses and further decompose those hypotheses into sub-hypotheses. A hypothesis is defined as a claim about the state of the word in which a crime took place and must be verified. Associated with each hypothesis is a series of evidence collection tests.

The following examples for a hypothesis framework come directly from Martignoni's paper "How to Reuse Knowledge about Forensic Investigations" [Ref 30]. An example of a hypothesis could be:

> H: Email account user@domain, registered by user Alice, has been used to send a harmful message M to user Bob. Alice was the author of M and its sender.

Hypothesis H can be decomposed into sub-hypotheses:

> H1: Alice has sent message M from her computer C
>
> H2: sendmail, the mail transfer agent installed on C, has been configured to use user@domain as the From: header
>
> H3: When M was sent (T), C has been in use
>
> H4: When M was sent (T), C was connected to the Internet

An example of an evidence collection test for hypothesis H3 might be:

> E1: Check if there are files modified, created, deleted, accessed at time T
>
> E2: Check if there are files that contain information about user activity (browser history, recent file list, etc)

E3: Check if there are files that contain information about system activity (events logs, application logs, etc) at time T

When this model is formalized it is represented as an acyclic case graph whose nodes are hypotheses and whose leaves are the evidence collecting tests attached to hypothesis nodes. These graphs can then be shared and interpreted as teaching aides to other forensic examiners as they identify hypotheses that fall in a common pattern. These graphs can also be used in a presentation setting, such as in a court room, where an examiner may have to describe his investigative process.

A formal model, such as the one being described, would be very relevant to the case report. The purpose of the case report is directly related to what is being represented in the case graph. A bookmark in Autopsy can be loosely used to represent an evidence-collecting test such as those described. For example, the investigator may add to the bookmark called “E2-User Activity” a cached web file, a shortcut link in the *Recent* folder, and a sent email file to signify that a subject was using the computer at a specific time. However, there needs to be some method for associating the bookmark “E2-User Activity” with the hypothesis it serves to verify.

In order for the bookmark features in Autopsy to successfully represent a case graph, additional functionality is needed with the creation and management of bookmarks. Future work to accomplish this could possibly involve the ability to nest bookmarks, define different types of bookmarks, formally describe bookmarks, and formally define a hypothesis. Some form of bookmark manager would be required. A case graph modeling the hypotheses and collections of evidence for an investigation could then be included in the case report.

Digital technologies will continue to evolve over time and therefore, the ways in which digital evidence is analyzed must evolve as well. Criminals will continue to figure out new ways of using their computers for illegal activities.

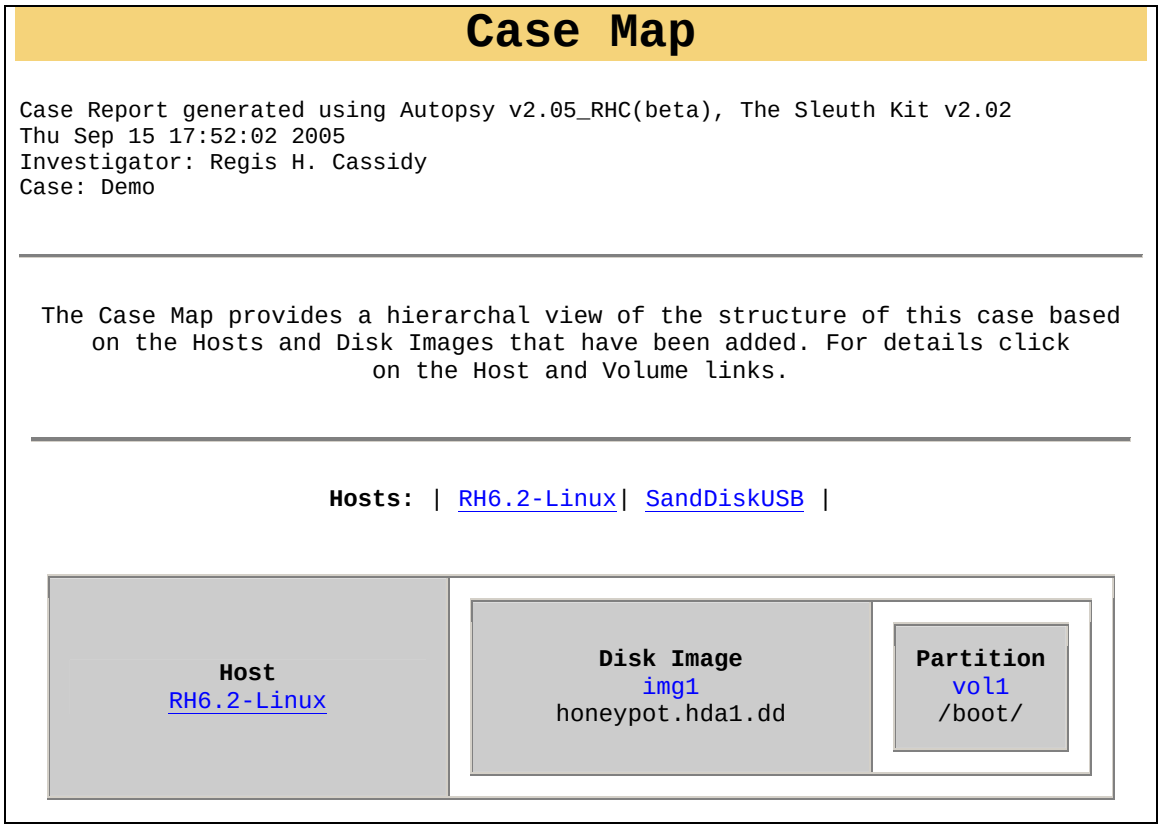
Forensic examiners will be right there following their tracks, learning about the new types of evidence they leave behind and ways to find it. As the art and science of digital forensics adjusts to changing technologies and crime, so will the tools used to process digital media. The development for automated case reports should continue to be synchronous with the development of methods for forensic analysis. Complete documentation and reports for analysis results will always be one of the most crucial elements of digital forensics.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. DEMO CASE REPORT

A sample of a case report generated from a Fedora Core 3 system running Autopsy version 2.05_RHC and Sleuth Kit version 2.02 is provided in this appendix. The details in this report are not taken from a real case. The disk image files used in this example were taken from the Honeynet Project Forensic Challenge [Ref 33] and from a SandDisk USB thumb drive. The thumb drive is not related to the Forensic Challenge images, but was added to show the use of multiple hosts in an Autopsy case. This case report demonstrates the analysis performed on these disk images, but is not intended to resemble a real investigation.

A. CASE MAP



	Disk Image img2 honeypot.hda5.dd	**Partition** vol2 /usr/
	Disk Image img3 honeypot.hda6.dd	**Partition** vol3 /home/
	Disk Image img4 honeypot.hda7.dd	**Partition** vol4 /var/
	Disk Image img5 honeypot.hda8.dd	**Partition** vol5 /
	Disk Image img6 honeypot.hda9.dd	**Partition** vol6 swap/

^top

Host SandDiskUSB	**Disk Image** img1 usbkey.dd	**Partition** vol1 H:/

^top

B. CASE DETAILS

Case Details

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:02 2005
Investigator: Regis H. Cassidy
Case: Demo

Case "Demo"

Case Directory: /home/regis/evidlock/Demo/

Description: This is a test case to demonstrate the case report feature in Autopsy v2.05_RHC(beta).
The first host add to this case is taken from the Honeynet.org Forensic Challenge (http://www.honeynet.org/fchallenge). The second host added to the case is from a SanDisk thumb drive.

Time Created: Thu Aug 25 12:54:50 2005

Investigators: Howard Fox
Regis H. Cassidy

Hosts: RH6.2-Linux
SandDiskUSB

Case History: /home/regis/evidlock/Demo/case.log

C. CASE HISTORY

Case History

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:02 2005
Investigator: Regis H. Cassidy
Case: Demo

Thu Aug 25 12:54:50 2005: Case Demo created by Regis H. Cassidy

Thu Aug 25 13:05:29 2005: Host RH6.2-Linux added to case by Regis H. Cassidy

Thu Aug 25 13:05:31 2005: Host RH6.2-Linux opened by Regis H. Cassidy

Thu Aug 25 13:15:36 2005: Case Demo opened

Thu Aug 25 13:20:18 2005: Case Demo opened

Thu Aug 25 13:31:39 2005: Case Demo opened

Thu Aug 25 13:31:41 2005: Host RH6.2-Linux opened by Regis H. Cassidy

Thu Aug 25 13:46:08 2005: Case Demo opened

Thu Aug 25 13:46:18 2005: Host RH6.2-Linux opened by Regis H. Cassidy

Thu Aug 25 15:08:13 2005: Case Demo opened

Thu Aug 25 15:08:16 2005: Host RH6.2-Linux opened by Regis H. Cassidy

Thu Aug 25 15:13:09 2005: Case Demo opened

Thu Aug 25 21:26:50 2005: Case Demo opened

Thu Aug 25 21:28:13 2005: Host RH6.2-Linux opened by Regis H. Cassidy

Thu Aug 25 21:30:58 2005: Bookmark 'Rootkit' added to case

...

D. HOST DETAILS

Host Details

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:02 2005
Investigator: Regis H. Cassidy
Case: Demo

Host "RH6.2-Linux"

Host Directory: /home/regis/evidlock/Demo/RH6.2-Linux/

Description: This honeypot was running a default server install of Red Hat Linux release 6.2 (kernel 2.2.14-5.0). On November 07, 2000, Lance Sptizner of the Honeynet Project reported that this machine was compromised.

Time Added: Thu Aug 25 13:05:29 2005

Investigator: Regis H. Cassidy

Host Timezone: 'CST6CDT'

Timeskew: '0'

Alert Hash Database:

Exclude Hash Database:

Images:

img1	honeypot.hda1.dd
img2	honeypot.hda5.dd
img3	honeypot.hda6.dd
img4	honeypot.hda7.dd
img5	honeypot.hda8.dd
img6	honeypot.hda9.dd

Host History: .../evidlock/Demo/RH6.2-Linux/logs/host.log

Timeline Files:

File Name	Type	Created	Investigator
body	body	Thu Aug 25 13:08:26 2005	Regis H. Cassidy
md5: EE09DFF31BD70390EF0B30B923B2EA43			
timeline.txt	timeline	Thu Aug 25 15:10:01 2005	Regis H. Cassidy

E. HOST HISTORY

History for Host RH6.2-Linux

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:02 2005
Investigator: Regis H. Cassidy
Case: Demo

Thu Aug 25 13:05:29 2005: Host RH6.2-Linux added to case Demo by Regis H. Cassidy

Thu Aug 25 13:05:31 2005: Host RH6.2-Linux opened by Regis H. Cassidy

Thu Aug 25 13:06:44 2005: Sym Linking image /home/regis/images/fchallenge/honeypot.hda1.dd into Demo:RH6.2-Linux

Thu Aug 25 13:06:44 2005: Image added: image img1 raw images/honeypot.hda1.dd by Regis H. Cassidy

Thu Aug 25 13:06:44 2005: Volume added: part vol1 img1 0 0 ext /boot by Regis H. Cassidy

Thu Aug 25 13:07:43 2005: Sym Linking image /home/regis/images/fchallenge/honeypot.hda5.dd into Demo:RH6.2-Linux

Thu Aug 25 13:07:43 2005: Image added: image img2 raw images/honeypot.hda5.dd by Regis H. Cassidy

Thu Aug 25 13:07:43 2005: Volume added: part vol2 img2 0 0 ext /usr by Regis H. Cassidy

Thu Aug 25 13:11:19 2005: Sym Linking image /home/regis/images/fchallenge/honeypot.hda6.dd into Demo:RH6.2-Linux

Thu Aug 25 13:11:19 2005: Image added: image img3 raw images/honeypot.hda6.dd by Regis H. Cassidy

F. IMAGE DETAILS

Image Details

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:03 2005
Investigator: Regis H. Cassidy
Case: Demo

Image "honeypot.hda1.dd" on Host "RH6.2-Linux"

Image Path: /home/regis/evidlock/Demo/RH6.2-Linux/images/honeypot.hda1.dd

Time Added: Thu Aug 25 13:06:44 2005

Investigator: Regis H. Cassidy

Volume ID: img1

MD5: 5329463466DB810B41D97BC9C34CDAF5

Partitions: vol1 honeypot.hda1.dd-0-0 /boot/ ext

G. PARTITION DETAILS

Partition Details

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:03 2005
Investigator: Regis H. Cassidy
Case: Demo

Partition "honeypot.hda1.dd-0-0" on host "RH6.2-Linux"

Mount Name: /boot/
File System: ext details
Volume ID: vol1
Sectors: This partition represents the entire image.

Extracted Files

ASCII Strings:

VolID	Name	Created	Investigator
vol9	honeypot.hda1.dd-0-0-ext.asc	Thu Aug 25 22:09:04 2005	Regis H. Cassidy

MD5: A2F1F4170BD860B54949D293D8EB20FE

Unicode Strings:

VolID	Name	Created	Investigator
vol10	honeypot.hda1.dd-0-0-ext.uni	Thu Aug 25 22:09:05 2005	Regis H. Cassidy

MD5: 0FCAB10E699B10E34E97365F179F39E5

Unallocated Fragments:

VolID	Name	Created	Investigator
vol11	honeypot.hda1.dd-0-0-ext.unalloc	Thu Aug 25 22:09:20 2005	Regis H. Cassidy

MD5: 1A99E177BA7EC9B5ED81EE796943699B

ASCII Strings of Unallocated:

VolID	Name	Created	Investigator
vol15	honeypot.hda1.dd-0-0-ext.unalloc-dls.asc	Thu Aug 25 22:13:58 2005	Regis H. Cassidy

MD5: E4C23762ED2823A27E62A64B95C024E7

H. FILE SYSTEM DETAILS

File System Details

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:03 2005
Investigator: Regis H. Cassidy
Case: Demo

FILE SYSTEM INFORMATION

File System Type: Ext2
Volume Name:
Volume ID: 617acac796d5059dd411b6b21ed94957

Last Written at: Sun Nov 5 07:33:31 2000
Last Checked at: Sat Nov 4 16:55:31 2000

Last Mounted at: Sun Nov 5 07:33:25 2000
Unmounted Improperly
Last mounted on:

Source OS: Linux
Dynamic Structure
InCompat Features: Filetype,
Read Only Compat Features: Sparse Super,

METADATA INFORMATION

Inode Range: 1 - 5040
Root Directory: 2
Free Inodes: 5016

CONTENT INFORMATION

Block Range: 0 - 20127
Block Size: 1024
Reserved Blocks Before Block Groups: 1
Free Blocks: 17005

BLOCK GROUP INFORMATION

Number of Block Groups: 3
Inodes per group: 1680
Blocks per group: 8192

Group: 0:
Inode Range: 1 - 1680
Block Range: 1 - 8192
Layout:
Super Block: 1 - 1
Group Descriptor Table: 2 - 2
Data bitmap: 3 - 3
...

I. INVESTIGATOR DETAILS

Investigator Details

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:03 2005
Investigator: Regis H. Cassidy
Case: Demo

Case Investigators: | Howard Fox | Regis H. Cassidy |

Investigator "Regis H. Cassidy"

Company/Agency: Naval Postgraduate School

Title: Research Associate

Address: 833 Dyer Road
Monterey, CA 93943-5118

Phone: 555-555-5555

Email: rhcassid@nps.edu

Comment: Example investigations are part of thesis research for the Center for Information Systems Security Studies and Research (CISR) at the Naval Postgraduate School.

Notes: **Host RH6.2-Linux**
/home/regis/evidlock/Demo/RH6.2-Linux/logs/Regis H. Cassidy.notes

History: **Host RH6.2-Linux**
/home/regis/evidlock/Demo/RH6.2-Linux/logs/Regis H. Cassidy.log

Command Log: **Host RH6.2-Linux**
/home/regis/evidlock/Demo/RH6.2-Linux/logs/Regis H. Cassidy.exec.log

J. INVESTIGATOR HISTORY

Regis H. Cassidy's History on RH6.2-Linux

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:03 2005
Investigator: Regis H. Cassidy
Case: Demo

Thu Aug 25 13:05:31 2005: Host RH6.2-Linux opened

Thu Aug 25 13:31:41 2005: Host RH6.2-Linux opened

Thu Aug 25 20:32:08 2005: Saving timeline data for Allocated Files, Unallocated Files, Unallocated Meta Data Structures for vol2, vol4, vol5, vol1, vol3 to output/body

Thu Aug 25 13:33:25 2005: body: Creating timeline for 11/6/2000-11/8/2000 (TZ: 'Etc/GMT-6') Password & Group File (vol5) Password Meta Address (26547) Group Meta Address (26553) to output/timeline.txt

Thu Aug 25 13:46:18 2005: Host RH6.2-Linux opened

Thu Aug 25 20:46:29 2005: Saving timeline data for Allocated Files, Unallocated Files, Unallocated Meta Data Structures for vol2, vol4, vol5, vol1, vol3 to output/body

Thu Aug 25 13:46:49 2005: body: Creating timeline for 11/6/2000-11/8/2000 (TZ: 'CST6CDT') Password & Group File (vol5) Password Meta Address (26547) Group Meta Address (26553) to output/timeline.txt

Thu Aug 25 13:47:43 2005: vol8: Viewing timeline for Nov 6-31 2000

Thu Aug 25 13:47:57 2005: vol8: Viewing timeline for Nov 8-31 2000

Thu Aug 25 13:49:10 2005: body: Creating timeline for 11/6/2000-11/9/2000 (TZ: 'CST6CDT') Password & Group File (vol5) Password Meta Address (26547) Group Meta Address (26553) to output/timeline.txt

...

K. INVESTIGATOR COMMAND HISTORY

Regis H. Cassidy's Command History on RH6.2-Linux

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:03 2005
Investigator: Regis H. Cassidy
Case: Demo

Thu Aug 25 13:06:20 2005: '/usr/local/sleuthkit/bin/img_stat' -t "/home/regis/images/fchallenge/honeypot.hda1.dd"

Thu Aug 25 13:06:20 2005: '/usr/local/sleuthkit/bin/fsstat' -t -i raw "/home/regis/images/fchallenge/honeypot.hda1.dd"

Thu Aug 25 13:06:43 2005: '/usr/local/sleuthkit/bin/img_stat' -t "/home/regis/images/fchallenge/honeypot.hda1.dd"

Thu Aug 25 13:06:43 2005: '/usr/local/sleuthkit/bin/dls' -f raw -e "/home/regis/images/fchallenge/honeypot.hda1.dd" | '/usr/local/sleuthkit/bin/md5'

Thu Aug 25 13:06:44 2005: '/usr/local/sleuthkit/bin/fsstat' -o 0 -i raw -f ext "/home/regis/images/fchallenge/honeypot.hda1.dd"

Thu Aug 25 13:06:44 2005: /bin/ln -s '/home/regis/images/fchallenge/honeypot.hda1.dd' '/home/regis/evidlock/ForensicChallenge/RH6.2-Linux/images/honeypot.hda1.dd'

Thu Aug 25 13:07:00 2005: '/usr/local/sleuthkit/bin/img_stat' -t "/home/regis/images/fchallenge/honeypot.hda5.dd"

Thu Aug 25 13:07:00 2005: '/usr/local/sleuthkit/bin/fsstat' -t -i raw "/home/regis/images/fchallenge/honeypot.hda5.dd"

Thu Aug 25 13:07:10 2005: '/usr/local/sleuthkit/bin/img_stat' -t "/home/regis/images/fchallenge/honeypot.hda5.dd"

Thu Aug 25 13:07:10 2005: '/usr/local/sleuthkit/bin/dls' -f raw -e "/home/regis/images/fchallenge/honeypot.hda5.dd" | '/usr/local/sleuthkit/bin/md5'

Thu Aug 25 13:07:43 2005: '/usr/local/sleuthkit/bin/fsstat' -o 0 -i raw -f ext "/home/regis/images/fchallenge/honeypot.hda5.dd"

...

L. LOG FILES

Log Files

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:03 2005
Investigator: Regis H. Cassidy
Case: Demo

Case History: /home/regis/evidlock/Demo/case.log

Host History:

RH6.2-Linux

/home/regis/evidlock/Demo/RH6.2-Linux/logs/host.log

Investigator History:

/home/regis/evidlock/Demo/RH6.2-Linux/logs/Regis H. Cassidy.log

Investigator Commands:

/home/regis/evidlock/Demo/RH6.2-Linux/logs/Regis H. Cassidy.exec.log

SandDiskUSB

/home/regis/evidlock/Demo/SandDiskUSB/logs/host.log

Investigator History:

/home/regis/evidlock/Demo/SandDiskUSB/logs/Howard Fox.log

Investigator Commands:

/home/regis/evidlock/Demo/SandDiskUSB/logs/Howard Fox.exec.log

/home/regis/evidlock/Demo/SandDiskUSB/logs/Regis H. Cassidy.exec.log

M. IMAGE INTEGRITY

Image Integrity

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:03 2005
Investigator: Regis H. Cassidy
Case: Demo

RH6.2-Linux

File System Images

honeypot.hda6.dd	A8437519727A72699215152155A3DE0A
honeypot.hda5.dd	A13854472EE4A7B0369669A68E454531
honeypot.hda7.dd	A71E39249F3DB55E8C96BAA72F7F25B1
honeypot.hda1.dd	5329463466DB810B41D97BC9C34CDAF5
honeypot.hda9.dd	1D1F5B6DDE0DB8D2A137BECC1E8B315E
honeypot.hda8.dd	19EC75E49D6A6172E36DF0C66D963061

Unallocated Data Files

honeypot.hda1.dd-0-0-ext.unalloc	1A99E177BA7EC9B5ED81EE796943699B

Strings Files

honeypot.hda7.dd-0-0-ext.asc	1E5B96102FACA87EEE33862CD8EE2A97
honeypot.hda1.dd-0-0-ext.uni	0FCAB10E699B10E34E97365F179F39E5

Timeline Data Files

body	EE09DFF31BD70390EF0B30B923B2EA43

Timeline Files

timeline.txt	3D7EF6289A9B4226637FF8D5532198D1

SandDiskUSB

File System Images

usbkey.dd	9261A82DC70A20B221DF66EC533DB943

Unallocated Data Files

Strings Files

usbkey.dd-0-0-fat16.asc	470979267E5B8F40ACF0E320F9AFA979
usbkey.dd-0-0-fat16.uni	C30AF61D0360B932C720A13A03568E55

N. CASE NOTES

Case Notes

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:03 2005
Investigator: Regis H. Cassidy
Case: Demo

Choose Bookmark

| <default> | eggdrop | Rootkit |

Rootkit	
RH6.2-Linux	Regis H. Cassidy
Tue Sep 6 23:24:50 2005 File: /usr/bin/ftp Volume: vol2 Meta: 16125 M-time: Sat Feb 5 00:02:09 2000 A-time: Wed Nov 8 08:29:27 2000 C-time: Sat Nov 4 18:58:28 2000 ftp client accessed. Hacker uses ftp to download rootkit to victim machine.	
RH6.2-Linux	Regis H. Cassidy
Tue Sep 6 23:24:55 2005 Directory: /usr/man/.Ci/ Volume: vol2 Meta: 109798 M-time: Wed Nov 8 08:56:08 2000 A-time: Wed Nov 8 08:56:57 2000 C-time: Wed Nov 8 08:56:08 2000 Hacker appears to unpack a tar archive to this "hidden" directory.	

default	
RH6.2-Linux	Regis H. Cassidy
Tue Sep 6 23:24:43 2005 File: /etc/hosts.deny Volume: vol5 Meta: 26217 M-time: Wed Nov 8 08:26:15 2000 A-time: Wed Nov 8 08:45:18 2000 C-time: Wed Nov 8 08:26:15 2000 The hosts.deny file has been zeroed out. This file normally provides tcp wrapper access controls.	

RH6.2-Linux	Regis H. Cassidy

Wed Sep 7 00:19:59 2005
File: /usr/bin/uptime
Volume: vol2 Meta: 17088
M-time: Tue Mar 7 12:03:26 2000
A-time: Wed Nov 8 08:25:53 2000
C-time: Sat Nov 4 19:03:17 2000

The hacker runs uptime. Most likely checking how long the machine has been running and is checking for the number of user logged in.

This event seems to be the first sign of the hacker's activity on the system.

eggdrop

RH6.2-Linux	Regis H. Cassidy

Tue Sep 6 23:25:35 2005
File: /honeypot.hda8.dd-dead-8133
Volume: vol5 Meta: 8133
M-time: Wed Nov 8 08:51:37 2000
A-time: Wed Nov 8 08:58:42 2000
C-time: Wed Nov 8 08:58:45 2000

Deleted file owned by user drosen. This is a tar archive containing an IRC bot name eggdrop.

untarred to default directory named w/ a space. Owner of contents is toro.

O. SEQUENCER EVENTS

Sequencer Events

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:03 2005
Investigator: Regis H. Cassidy
Case: Demo

Choose Bookmark

All Events
| default | eggdrop | Rootkit |

Bookmark	Date & Time	Source	Event & Note	Host & Investigator
Rootkit	**Nov 08, 2000 08:29:27**	/usr/bin/ftp ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for GNU/Linux 2.0.0, dynamically linked (uses shared libs), stripped	[A-Time]ftp client accessed. Hacker uses ftp to download rootkit to victim machine.	RH6.2-Linux Regis H. Cassidy
Rootkit	**Nov 08, 2000 08:56:08**	/usr/man/.Ci/ data	[M-Time]Hacker appears to unpack a tar archive to this "hidden" directory.	RH6.2-Linux Regis H. Cassidy

P. KEYWORD SEARCHES

Keyword Searches

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:03 2005
Investigator: Regis H. Cassidy
Case: Demo

Keyword	Hits	Type	vol		Host
Regis	53	ascii	vol1	usbkey.dd-0-0	SandDiskUSB
Regis	40	unicode	vol1	usbkey.dd-0-0	SandDiskUSB
root	173	ascii	vol6	honeypot.hda9.dd-0-0	RH6.2-Linux
tpack	57	ascii	vol5	honeypot.hda8.dd-0-0	RH6.2-Linux

57 occurrences of tpack were found
Search Options:
  ASCII
  Case Insensitive

Fragment 8964
Inode: 2067
File: inode not currently used
1: 359 (.1.6+tPACK 2.3")
2: 409 (.1.6+tPACK 2.3")

Fragment 8995
Inode: 2067
File: inode not currently used
3: 841 ("pid.tpack", bo)

Fragment 33063
4: 56 (tpack23-ef)

Fragment 33533
Inode: 8133
File: inode not currently used
5: 871 (.1.6+tPACK 2.3")
6: 921 (.1.6+tPACK 2.3")
...

Q. FILE CATEGORIES

File Categories

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:03 2005
Investigator: Regis H. Cassidy
Case: Demo

	RH6.2-Linux				SandDiskUSB
	/usr/	/home/	/var/	/	H:/
Files	24147	157	213	7184	117
Allocated	22407	153	203	7059	89
Unallocated	1740	4	10	125	28
Files Skipped	1822	23	151	5830	37
Non-Files	1822	23	151	5830	37
'ignore' category	0	0	0	0	0
Extension Mismatches	3604	0	31	92	10
Categories	22325	134	62	1354	80
archive	2203	0	8	9	0
audio	0	0	0	0	0
compress	524	0	0	0	4
crypto	2	0	0	0	0
data	460	1	9	39	14
disk	0	0	0	0	0
documents	236	0	0	0	39
exec	2222	9	3	771	7
images	229	112	0	0	5
system	0	0	0	0	3
text	13264	12	42	522	5
unknown	3185	0	0	13	3
video	0	0	0	0	0

R. FILE CATEGORY RESULTS

File Categories: mismatch

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:03 2005
Investigator: Regis H. Cassidy
Case: Demo

RH6.2-Linux /usr/

Extension Mismatch

/usr/doc/glibc-2.1.3/Changes.threads
ASCII English text (Ext: threads)
Image: /home/regis/evidlock/ForensicChallenge/RH6.2-Linux/images/honeypot.hda5.dd Inode: 30806

/usr/doc/glibc-2.1.3/README.crypt
ASCII English text (Ext: crypt)
Image: /home/regis/evidlock/ForensicChallenge/RH6.2-Linux/images/honeypot.hda5.dd Inode: 30814

/usr/doc/glibc-2.1.3/README.db2
ASCII English text (Ext: db2)
Image: /home/regis/evidlock/ForensicChallenge/RH6.2-Linux/images/honeypot.hda5.dd Inode: 30815

/usr/doc/glibc-2.1.3/README.db2.mutex
ASCII English text (Ext: mutex)
Image: /home/regis/evidlock/ForensicChallenge/RH6.2-Linux/images/honeypot.hda5.dd Inode: 30816

/usr/doc/glibc-2.1.3/README.hesiod
ASCII English text (Ext: hesiod)
Image: /home/regis/evidlock/ForensicChallenge/RH6.2-Linux/images/honeypot.hda5.dd Inode: 30817

...

S. PICTURE THUMBNAILS

Picture Thumbnails

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:05 2005
Investigator: Regis H. Cassidy
Case: Demo

To view thumbnails be sure to create a sequencer event for the 'interesting' images.

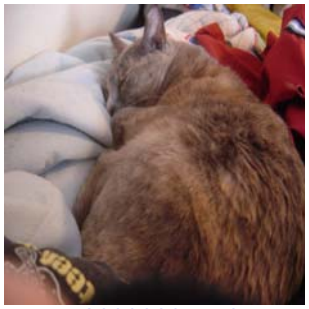
1

smallliz.jpg
[A-Time]
Jul 27, 1999 16:50:27

dave.gif
[A-Time]
Jul 27, 1999 16:50:27

cramps.gif
[M-Time]
Jul 27, 1999 16:50:27

2



DSC00939.JPG
[A-Time]
Jul 29, 2005 00:00:00

cat.gif
[C-Time]
Nov 04, 2000 19:01:11

T. PICTURE DETAILS

Picture Details

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:05 2005
Investigator: Regis H. Cassidy
Case: Demo

/usr/doc/libtiff-devel-3.5.4/images/cat.gif
GIF image data, version 89a, 113 x 146

[C-Time] Nov 04, 2000 19:01:11

Bookmarked by Regis H. Cassidy
Bookmark Name: IMG-Animals

Note:
Nice Kitty!

U. INDIVIDUAL REPORTS

Individual Reports

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:05 2005
Investigator: Regis H. Cassidy
Case: Demo

Individual reports must be saved to /home/regis/evidlock/Demo/*host-dir*/reports

Host RH6.2-Linux

- egg.log
- list-of-eggdrop-files.txt
- tpack-eggdrop-install.txt
- usr.man..Ci.md5.txt

V. DETAILS OF AN INDIVIDUAL REPORT

Autopsy string Fragment Report

Case Report generated using Autopsy v2.05_RHC(beta), The Sleuth Kit v2.02
Thu Sep 15 17:52:05 2005
Investigator: Regis H. Cassidy
Case: Demo

Autopsy string Fragment Report

GENERAL INFORMATION

Fragment: 34689
Fragment Size: 1024

Pointed to by Inode: 8133
Pointed to by files:
inode not currently used

MD5 of raw Fragment: c25becd012804439037fb606a4151327
MD5 of string output: 3120de20168726f3c0902ca44f5a2fbd

Image: '/home/regis/evidlock/ForensicChallenge/RH6.2-Linux/images/honeypot.hda8.dd'

Offset: Full image
File System Type: ext

Date Generated: Thu Aug 25 23:53:02 2005

Investigator: Regis H. Cassidy

CONTENT

```
unset HIST
chmod a-w ~/.bash_history
./configure --silent
make eggdrop
mv eggdrop p
rm -rf src
rm install
gcc encrypt.c -o encrypt
rm *.c
rm config*
rm lush*
rm Make*
```

```
rm *.h > /dev/null
rm DEBUG*
chmod 700 run
echo " "
echo "Completed installation of tpack version 2.3"
/config.log
100644
1750
144
4354 7165771137 11574
ustar
toro
users

------------------------------------------------------------

VERSION INFORMATION

Autopsy Version: 2.05_RHC(beta)

The Sleuth Kit Version: 2.02
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. SOURCE CODE

The source code for the case report module in Autopsy version 2.05_RHC (beta), *Reports.pm*, is provided in this appendix. The source code for the remaining Autopsy modules is also provided if any part of them was modified for the work done in this thesis. Added or modified lines of code have been identified using a *diff* [Ref 34] program.

A. REPORTS.PM

```
#
# $Date: 2005/09/16 08:31:52 $
#
# Generate case reports for the investigation
#
# Brian Carrier [carrier@sleuthkit.org]
# Regis Friend Cassidy [rfriendcassidy@gmail.com]
# Copyright (c) 2001-2005 by Brian Carrier.  All rights reserved
#
# This file is part of the Autopsy Forensic Browser (Autopsy)
#
# Autopsy is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# Autopsy is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with Autopsy; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
#
# THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR IMPLIED
# WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF
# MERCHANTABILITY AND FITNESS FOR ANY PARTICULAR PURPOSE.
# IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
# INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
# (INCLUDING, BUT NOT LIMITED TO, LOSS OF USE, DATA, OR PROFITS OR
# BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
# WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
# OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
# ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

package Reports;
use File::Copy;

#Main Frame Views
```

```
$Reports::FRAME        = 0;
$Reports::NAV_BAR      = 1;

#Case Views
$Reports::CASE_MAP     = 2;
$Reports::CASE_DETAILS = 3;
my $CASE_MAX = 5;

#Investigator Views
$Reports::INV_DETAILS  = 8;
$Reports::INV_NOTES    = 9;
my $INV_MAX = 9;

#Host and Image Views
$Reports::HOST_DETAILS = 10;
$Reports::IMG_DETAILS  = 11;
$Reports::PART_DETAILS = 12;
$Reports::FS_DETAILS   = 13;
my $IMG_MAX = 13;

#Notes Views
$Reports::NOTES_FRAME  = 14;
$Reports::NOTES_BMARKS = 15;
$Reports::NOTES_ALL    = 16;
my $NOTES_HTML_MAX = 16;

#Sequencer Events Views
$Reports::SEQ_FRAME    = 17;
$Reports::SEQ_BMARKS   = 18;
$Reports::SEQ_EVENTS   = 19;
my $SEQ_MAX = 19;

#Keyword Search Views
$Reports::KWORD_FRAME  = 20;
$Reports::KWORD_LIST   = 21;
$Reports::KWORD_RESULTS = 22;
my $KWORD_MAX = 22;

#File Views
$Reports::FILE_CAT_SUM = 23;
$Reports::FILE_CAT = 24;
$Reports::THUMBS = 25;
$Reports::PIC_DETAILS = 26;
my $FILE_MAX = 26;

#Individual Reports View
$Reports::INDV_REP = 27;
my $INDV_REP_MAX = 27;

#Case Audit Views
$Reports::LOGS = 28;
$Reports::IMG_INTEG = 29;
my $CASE_AUDIT_MAX = 29;

#Various functions
$Reports::PRINT_2HTML   = 30;
$Reports::SAVE_REPORT  = 31;

my $SAVE = 0; # 0 = live mode, 1 = save to html files

$BGCOLOR = "#F5D37A"; #background color for tables
#--------------------------------------
```

```
# HTML directories for saving Report
#-------------------------------------
$PICDIR      = "pict";
$CASEDIR     = "case";
$INVDIR      = "investigators";
$HOSTDIR     = "hosts";
$IMGDIR      = "images";
$PARTDIR     = "parts";
$NOTESDIR    = "notes";
$EVENTSDIR   = "events";
$KWORDDIR    = "kwordSearch";
$FILESDIR    = "files";
$THUMBSDIR   = "pics";
$INDVREPDIR  = "indv_reports";
$AUDITDIR    = "caseaudit";

#-------------------------------------
# HTML File names for saving Report
#-------------------------------------
$INDEX_HTML    = "index.html";
$NAV_HTML      = "nav_bar.html";
$DETAILS_HTML  = "details.html";
$HIST_HTML     = "history.html";
$ALERT_HTML    = "alertdb.html";
$EXCLUDE_HTML  = "excludedb.html";
$COM_HTML      = "commands.html";
$NOTES_HTML    = "notes.html";
$BMARKS_HTML   = "bookmarks.html";
$EVENTS_HTML   = "seqevents.html";
$MAP_HTML      = "map.html";
$LIST_HTML     = "list.html";
$KWORDS_HTML   = "results.html";
$FILECAT_HTML  = "file_categories.html";
$THUMBS_HTML   = "thumbs.html";
$MD5S_HTML     = "imgintegrity.html";
$LOGS_HTML     = "caselogs.html";

sub main {
  $aut_ver = $::VER;
  $tsk_ver = ::get_tskver();

  # By default, show the main frame
  $Args::args{'view'} = $Args::enc_args{'view'} = $Reports::FRAME
    unless (exists $Args::args{'view'});

  # need to set case_dir for logging
  Args::check_case();
  $::case_dir = "$::LOCKDIR/" . Args::get_case() . "/";
  $::case_dir =~ s/\/\//\//g;       #replace double slashes with a single slash

  $REPORT_URL =
"$::PROGNAME?mod=$::MOD_REPORTS&case=$Args::args{'case'}&inv=$Args::args{'inv'}
";
  $ROOTDIR     = "$::case_dir/casereports";

  Args::check_view();
  my $view = Args::get_view();

  # Report functions
  if ($view <= $CASE_MAX) {
    if ($view == $Reports::FRAME) {
      return frame();
```

```
  }
  elsif ($view == $Reports::NAV_BAR) {
    return nav_bar();
  }
  elsif ($view == $Reports::CASE_MAP) {
    return case_map();
  }
  elsif ($view == $Reports::CASE_DETAILS) {
    return case_details();
  }
}

elsif ($view <= $INV_MAX) {
  if ($view == $Reports::INV_DETAILS) {
    return inv_details();
  }
  elsif ($view == $Reports::INV_NOTES) {
    return inv_notes($Args::args{'luinv'});
  }
}

elsif ($view <= $IMG_MAX) {
  if ($view == $Reports::HOST_DETAILS) {
    return host_details();
  }
  elsif ($view == $Reports::IMG_DETAILS) {
    return img_details();
  }
  elsif ($view == $Reports::PART_DETAILS) {
    return part_details();
  }
  elsif ($view == $Reports::FS_DETAILS) {
    return fs_details();
  }
}

elsif ($view <= $NOTES_HTML_MAX) {
  if ($view == $Reports::NOTES_FRAME) {
    return notes_frame();
  }
  elsif ($view == $Reports::NOTES_BMARKS) {
    return notes_bmarks();
  }
  elsif ($view == $Reports::NOTES_ALL) {
    return notes_all();
  }
}

elsif ($view <= $SEQ_MAX) {
  if ($view == $Reports::SEQ_FRAME) {
    return seq_frame();
  }
  elsif ($view == $Reports::SEQ_BMARKS) {
    return seq_bmarks();
  }
  elsif ($view == $Reports::SEQ_EVENTS) {
    return seq_events($Args::args{'bmark'});
  }
}

elsif ($view <= $KWORD_MAX) {
  if ($view == $Reports::KWORD_FRAME) {
```

```
      return kword_frame();
    }
    elsif ($view == $Reports::KWORD_LIST) {
      return kword_list();
    }
    elsif ($view == $Reports::KWORD_RESULTS) {
      return kword_results();
    }
  }

  elsif ($view <= $FILE_MAX) {
    if ($view == $Reports::FILE_CAT_SUM) {
      return file_cat_sum();
    }
    elsif ($view == $Reports::FILE_CAT) {
      return file_cat();
    }
    elsif ($view == $Reports::THUMBS) {
      return thumbs();
    }
    elsif ($view == $Reports::PIC_DETAILS) {
      return pic_details();
    }
  }

  elsif ($view <= $INDV_REP_MAX) {
    if ($view == $Reports::INDV_REP) {
      return indv_rep();
    }
  }

  elsif ($view <= $CASE_AUDIT_MAX) {
    if ($view == $Reports::LOGS) {
      return logs();
    }
    elsif ($view = $Reports::IMG_INTEG) {
      return img_integ();
    }
  }
  elsif ($view == $Reports::PRINT_2HTML) {
    return print_2html($Args::args{'show'});
  }

  elsif ($view == $Reports::SAVE_REPORT) {
    $SAVE = 1;
     return save_report();
  }
}

# Writes the Case Report to a specified directory
sub save_report() {
  Print::print_html_header("Saving Report for Case $Args::args{'case'}");

  #log activity
  Print::log_case_info("Case Report $Args::args{'rname'} saved by
$Args::args{'inv'}");

  print "Saving Report \"$Args::args{'rname'}\" for Case
$Args::args{'case'}<p>";

  #save stdout
  open (SAVEOUT, ">&STDOUT");
```

```
mk_htmldir("$ROOTDIR");
my $SAVEDIR = "$ROOTDIR/$Args::args{'rname'}";
mk_htmldir($SAVEDIR);

#copy autopsy report graphic
mk_htmldir("$SAVEDIR/$PICDIR");

my $logo = "$SAVEDIR/$PICDIR/report_logo.jpg";
unless (-e "$logo") {
  copy("$::PICTDIR/report_logo.jpg", "$logo");
}

#case dir
mk_htmldir("$SAVEDIR/$CASEDIR");

#--------------------------
# MAIN INDEX file
#--------------------------
#redirect stdout to file
mk_htmlfile("$SAVEDIR/$INDEX_HTML");
frame();

#--------------------------
# NAV BAR File
#--------------------------
mk_htmlfile("$SAVEDIR/$NAV_HTML");
nav_bar();

print SAVEOUT "<p>Saving Case Information files</p>\n";

#--------------------------
# CASE DETAILS File
#--------------------------
mk_htmlfile("$SAVEDIR/$CASEDIR/$DETAILS_HTML");
case_details();

# case log
mk_htmlfile("$SAVEDIR/$CASEDIR/$HIST_HTML");
print_2html("caselog");

#--------------------------
# CASE MAP File
#--------------------------
mk_htmlfile("$SAVEDIR/$CASEDIR/$MAP_HTML");
case_map();

my @hosts = get_hosts();
if (scalar @hosts > 0) {
  #hosts dir
  mk_htmldir("$SAVEDIR/$HOSTDIR");

  foreach my $h (@hosts) {
    $::host_dir = "$::case_dir" . "$h" . "/";
    $Args::args{'host'} = $Args::enc_args{'host'} = $h;
    Caseman::read_host_config();

    #hostname dir
    #remove tainting
    $h = $1 if ($h =~ /($::REG_HOST)/);
    mk_htmldir("$SAVEDIR/$HOSTDIR/$h");
```

```
#--------------------------
# HOSTS DETAILS File
#--------------------------
mk_htmlfile("$SAVEDIR/$HOSTDIR/$h/$DETAILS_HTML");
host_details();

#alert hash db
if (-e "$Caseman::alert_db") {
 mk_htmlfile ("$SAVEDIR/$HOSTDIR/$h/$ALERT_HTML");
 print_2html("alertdb");
}
#exclude hash db
if (-e "$Caseman::exclude_db") {
 mk_htmlfile("$SAVEDIR/$HOSTDIR/$h/$EXCLUDE_HTML");
 print_2html("excludedb");
}
#Host Log
mk_htmlfile("$SAVEDIR/$HOSTDIR/$h/$HIST_HTML");
print_2html("hostlog");

#read image files
my @imgs;
if (scalar(keys %Caseman::vol2ftype) !=0) {
 foreach my $i (keys %Caseman::vol2cat) {
   if ($Caseman::vol2cat{$i} eq "image") {
     push @imgs, $i;
   }
 }
 #image dir
 my $imgdir = "$SAVEDIR/$HOSTDIR/$h/$IMGDIR";
 mk_htmldir("$imgdir");

 foreach my $i (@imgs) {
   $i = $1 if ($i =~ /($::REG_VNAME)/);
   #image name dir
   my $voldir = "$imgdir/$i";
   mk_htmldir("$voldir");

   #--------------------------
   # IMAGE DETAILS File
   #--------------------------
   mk_htmlfile("$voldir/$DETAILS_HTML");
   $Args::args{'img'} = $Args::enc_args{'img'} = $i;
   img_details();

   #Partitions
   my $partcount = scalar @{$Caseman::vol2part{$i}};

   if ($partcount > 0) {
     #part dir
     my $partdir = "$voldir/$PARTDIR";
     mk_htmldir("$partdir");

     foreach my $p (@{$Caseman::vol2part{$i}}) {
       #partname dir
       $p = $1 if ($p =~ /($::REG_VNAME)/);
       my $voldir = "$partdir/$p";
       mk_htmldir("$voldir");

       #------------------------
       # PARTITION DETAILS File
       #------------------------
```

```
          mk_htmlfile("$voldir/$DETAILS_HTML");
          $Args::args{'part'} = $Args::enc_args{'part'} = $p;
          part_details();
          #-----------------------
          # FILE SYSTEM DETAILS
          #-----------------------
          mk_htmlfile("$voldir/fsdetails-$p.html");
          $Args::args{'vol'} = $Args::enc_args{'vol'} = $p;
          fs_details();
        }
      }
    }
    }
  }
}

#investigator dir
my @inv = Caseman::read_invest();
if ( inv_unknown() ) {
  push @inv, "unknown";
}

mk_htmldir("$SAVEDIR/$INVDIR");
if (scalar @inv > 0) {
  #individual investigator's directories
  foreach my $i (@inv) {
    mk_htmldir("$SAVEDIR/$INVDIR/$i");
    #-------------------------
    # INVESTIGATOR DETAILS
    #-------------------------
    mk_htmlfile("$SAVEDIR/$INVDIR/$i/$DETAILS_HTML");
    $Args::args{'lu_inv'} = $Args::enc_args{'lu_inv'} = $i;
    inv_details();

    #investigator's notes
    mk_htmlfile("$SAVEDIR/$INVDIR/$i/$NOTES_HTML");
    inv_notes($i);

    #investigator's history
    foreach my $h (@hosts) {
     if (-e "$::case_dir$h/$::LOGDIR/$i.log") {
       mk_htmlfile("$SAVEDIR/$INVDIR/$i/$h$HIST_HTML");
       $Args::args{'host'} = $Args::enc_args{'host'} = $h;
       $Args::args{'luinv'} = $Args::enc_args{'luinv'} = $i;
       print_2html("invlog");
     }
    }
    #investigator's history
    foreach my $h (@hosts) {
     if (-e "$::case_dir$h/$::LOGDIR/$i.exec.log") {
       mk_htmlfile("$SAVEDIR/$INVDIR/$i/$h$COM_HTML");
       $Args::args{'host'} = $Args::enc_args{'host'} = $h;
       $Args::args{'luinv'} = $Args::enc_args{'luinv'} = $i;
       print_2html("invcomm");
     }
    }
  }
}

print SAVEOUT "<p>Saving Case Audit files</p>\n";
mk_htmldir("$SAVEDIR/$AUDITDIR");
#-------------------------
```

```
# LOG FILES
#--------------------------
mk_htmlfile("$SAVEDIR/$AUDITDIR/$LOGS_HTML");
logs();
#--------------------------
# IMG INTEGRITY
#--------------------------
mk_htmlfile("$SAVEDIR/$AUDITDIR/$MD5S_HTML");
img_integ();

print SAVEOUT "<p>Saving Analysis Information</p>\n";
#--------------------------
# BOOKMARKS (NOTES)
#--------------------------
mk_htmldir("$SAVEDIR/$NOTESDIR");

mk_htmlfile("$SAVEDIR/$NOTESDIR/$INDEX_HTML");
notes_frame();

mk_htmlfile("$SAVEDIR/$NOTESDIR/$BMARKS_HTML");
notes_bmarks();

mk_htmlfile("$SAVEDIR/$NOTESDIR/$NOTES_HTML");
notes_all();

#--------------------------
# SEQUENCER EVENTS
#--------------------------
mk_htmldir("$SAVEDIR/$EVENTSDIR");

mk_htmlfile("$SAVEDIR/$EVENTSDIR/$INDEX_HTML");
seq_frame();

mk_htmlfile("$SAVEDIR/$EVENTSDIR/$BMARKS_HTML");
seq_bmarks();

#timeline of all events
mk_htmlfile("$SAVEDIR/$EVENTSDIR/$EVENTS_HTML");
seq_events();

#timeline of non-bookmarked events
mk_htmlfile("$SAVEDIR/$EVENTSDIR/default-events.html");
seq_events("");

#create timelines of individual bookmarks
my @bookmarks = Notes::read_bookmarks();
foreach my $b (@bookmarks) {
  mk_htmlfile("$SAVEDIR/$EVENTSDIR/$b-events.html");
  seq_events($b);
}

#--------------------------
# KEYWORD SEARCHES
#--------------------------
mk_htmldir("$SAVEDIR/$KWORDDIR");

mk_htmlfile("$SAVEDIR/$KWORDDIR/$INDEX_HTML");
kword_frame();

mk_htmlfile("$SAVEDIR/$KWORDDIR/$LIST_HTML");
kword_list();
```

```
#generate results pages
my %searches = get_searches();
mk_htmlfile("$SAVEDIR/$KWORDDIR/choose.html");
kword_results(); #creates the generic page because no args are set
foreach my $srch (sort keys %searches) {
  foreach my $result (@{ $searches{$srch} } ) {
    my $type = $result->[1];
    my $host = $result->[3];
    my $sname = $result->[4];
    my $vol = $result->[5];
    my $fname = $result->[6];

    mk_htmlfile("$SAVEDIR/$KWORDDIR/$srch-$sname-$type.html");
    $Args::args{'host'} = $Args::enc_args{'host'} = $host;
    $Args::args{'vol'} = $Args::enc_args{'vol'} = $vol;
    $Args::args{'fname'} = $Args::enc_args{'fname'} = $fname;
    $::host_dir = "$::case_dir$host/";
    Caseman::read_host_config();
    kword_results();
  }
}

#-------------------------
# FILE CATEGORIES
#-------------------------
mk_htmldir("$SAVEDIR/$FILESDIR");

mk_htmlfile("$SAVEDIR/$FILESDIR/$FILECAT_HTML");
file_cat_sum();

#scan for sorter directories in each host
foreach my $h (sort @hosts) {
  my $output_dir = "$::case_dir$h/$::DATADIR/";
  opendir OUTPUT, $output_dir
    or die "Can't open directory $output_dir";

  foreach my $sort_dir (readdir OUTPUT) {
    next unless ($sort_dir =~ /^sorter\-(vol\d+)$/);
    if (-e "$output_dir/$sort_dir/index.html") {
     my $vol = $1;
     # parse the index.html file
     open INDEX, "<$output_dir/$sort_dir/index.html"
       or die "Can't open sorter index file ($sort_dir/index.html)";

     my $catfile;
     while (<INDEX>) {
       next unless(/^\s*(<.*>)?(<.*>)?([\w\s\'\-]+)(<.*>)?\s\((\d+)\)$/);
       if ($3 eq "Extension Mismatches") {
         $catfile = "mismatch";
       }
       else {
         $catfile = $3;
       }
       if ($5 > 0 && -e "$output_dir/$sort_dir/$catfile.html") {
         $Args::args{'host'} = $Args::enc_args{'host'} = $h;
         $Args::args{'vol'} = $Args::enc_args{'vol'} = $vol;

         $Args::args{'cat'} = $Args::enc_args{'cat'} = $catfile;

         mk_htmlfile("$SAVEDIR/$FILESDIR/$h-$vol-$catfile.html");
         file_cat();
       }
```

```
       }
       close(INDEX);
      }
    }
    close(OUTPUT);
  }

  #-------------------------
  # PICTURE THUMBNAILS
  #-------------------------
  mk_htmldir("$SAVEDIR/$FILESDIR/$THUMBSDIR");

  #export all bookmarked pictures
  my %events = get_seq_events();

  foreach my $i (@{ $events{'sorted'} }) {
    if (($events{filetype}[$i] =~ /image data/) || ($events{filetype}[$i] =~
/PC bitmap data/)) {
      $Args::args{'host'} = $events{host}[$i];
      $::host_dir = $::case_dir . $events{host}[$i] . "/";
      Caseman::read_host_config();

      my $mnt = $Caseman::vol2mnt{$events{vol}[$i]};
      my $ftype   = $Caseman::vol2ftype{$events{vol}[$i]};
      my $img     = $Caseman::vol2path{$events{vol}[$i]};
      my $offset  = $Caseman::vol2start{$events{vol}[$i]};
      my $imgtype = $Caseman::vol2itype{$events{vol}[$i]};
      my $dir = $1 if ($events{fname}[$i] =~ /^$mnt(.*)$/);
      my $filename = $1 if ($dir =~ /\/?([^\/]*)$/);
      my $note = "";
      my $time = "";
      if ($events{note}[$i] =~ /^(\[.?\-Time\])(.*)$/) {
       $time = $1 if (defined $1);
       $note = $2 if (defined $2);
      }
      my $date =
"$::d2m[$events{mon}[$i]] $events{day}[$i], $events{year}[$i]"
             . " $events{hour}[$i]:$events{min}[$i]:$events{sec}[$i]";

      mk_htmlfile("$SAVEDIR/$FILESDIR/$THUMBSDIR/$events{meta}[$i]-$filename");
      local *OUT;
      Exec::exec_pipe(*OUT,
       "'$::TSKDIR/icat' -f $ftype -r -o $offset -i $imgtype $img
$events{meta}[$i]"
      );
      while ($_ = Exec::read_pipe_line(*OUT)) {
       print "$_";
      }
      close(OUT);

      #pic details page
      $Args::args{'vol'} = $events{vol}[$i];
      $Args::args{'meta'} = $events{meta}[$i];
      $Args::args{'dir'} = $dir;
      $Args::args{'filetype'} = $events{filetype}[$i];
      $Args::args{'date'} = $date;
      $Args::args{'luinv'} = $events{invest}[$i];
      $Args::args{'note'} = $note;
      $Args::args{'bookmark'} = $events{bmark}[$i];
      $Args::args{'time'} = $time;
      mk_htmlfile("$SAVEDIR/$FILESDIR/$THUMBSDIR/$events{meta}[$i]-
$filename.html");
```

```
      pic_details();
    }
  }
  #thumbs html page
  mk_htmlfile("$SAVEDIR/$FILESDIR/$THUMBS_HTML");
  thumbs();

  #-------------------------
  # INDIVIDUAL REPORTS
  #-------------------------
  mk_htmldir("$SAVEDIR/$INDVREPDIR");
  mk_htmlfile("$SAVEDIR/$INDVREPDIR/$INDEX_HTML");
  indv_rep();

  my %reports = get_reports();
  foreach my $host (sort keys %reports) {
    $host =~ /($::REG_HOST)/;
    $host = $1;
    mk_htmldir("$SAVEDIR/$INDVREPDIR/$host/");
    foreach my $rep (sort @{ $reports{$host} }) {
      $rep =~ /($::REG_FILE)/;
      $rep = $1;
      $Args::args{'host'} = $host;
      $Args::args{'report'} = $rep;
      mk_htmlfile("$SAVEDIR/$INDVREPDIR/$host/$rep.html");
      print_2html('report');
    }
  }

  #restore stdout
  open(STDOUT, ">&SAVEOUT");

  print "<p></p><p>Case Report $Args::args{'rname'} saved.<br>\n"
      . "You can view the case report by loading $SAVEDIR/$INDEX_HTML<br>\n"
      . "in a web browser</p><p></p><p>Close this windows to return to
Autopsy</p>";

  Print::print_html_footer();
  return 0;
}

# Generate the html frames page for nav bar and main content
sub frame {
  unless ((exists $Args::args{'inv'}) && ($Args::args{'inv'} ne "")) {
    my @invs = Caseman::read_invest();

    if (scalar @invs == 0) {
      $Args::args{'inv'} = $Args::enc_args{'inv'} = 'unknown';
    }

    else {
      Print::print_html_header("Missing Investigator");
      print "<br>An investigator must be selected<p>\n"
       . "<form action=\"$::PROGNAME\" method=\"get\">\n"
         . "<input type=\"hidden\" name=\"mod\" value=\"$::MOD_REPORTS\">\n"
           . "<input type=\"hidden\" name=\"view\" value=\"$Reports::FRAME\">\n"
             . Args::make_hidden();

      print "Select one of the following:";
      print "<select name=\"inv\" size=\"1\">\n";

      print "<option value=\"\" selected>Select One" . "</option>\n";
```

```
      foreach my $i (@invs) {
       print "<option value=\"$i\">$i</option>\n";
      }
      print "</select><p>\n"
       . "<input type=\"image\" src=\"pict/but_ok.jpg\" alt=\"Ok\" "
             . "width=43 height=20 border=\"0\">\n"
              . "</form>\n";

      Print::print_html_footer();
      return 0;
    }
  }

  #if (!SAVE) {
  #  Print::log_case_info("Case Report for $Args::args{'case'} generated by
$Args::args{'inv'}");
  #}

  # begin printing html frames
  Print::print_html_header_frameset("Autopsy Case Report for
$Args::args{'case'}", $SAVE);
  print "<frameset cols=\"200,100%\" framespacing=\"0\" border=\"0\">\n";

  if ($SAVE) {
    print "<frame src=\"$NAV_HTML\">\n"
       . "<frame src=\"$CASEDIR/$MAP_HTML\" name=\"content\">\n</frameset>\n";
  }
  else {
    print "<frame src=\"$REPORT_URL&view=$Reports::NAV_BAR\">\n"
       . "<frame src=\"$REPORT_URL&view=$Reports::CASE_MAP\" "
       . "name=\"content\">\n</frameset>\n";
  }
  print <<EOF;
<NOFRAMES>
  <center>
    Autopsy requires a browser that supports frames.
  </center>
</NOFRAMES>
EOF

  Print::print_html_footer_frameset();
  return 0;
}

# Generate the navigation side-bar
sub nav_bar() {
  Print::print_report_nav_html_header("Autopsy Case Report Navigation", $SAVE);
  print "<table width=\"100%\" cellpadding=\"5\" border=\"0\">\n"
      . "<tr><td align=\"left\" valign=\"top\">\n"
      . "<p align=\"center\"><img src=\"pict/report_logo.jpg\" alt=\"Autopsy
Case Report Logo\"><br>\n"
      . "<strong>Autopsy Case Report</strong></p>\n";

  if (! $SAVE) {
    print "<form name=\"save_report\" method=\"get\" action=\"$::PROGNAME\"
target=\"main\">\n"
       . "<input type=\"hidden\" name=\"mod\" value=\"$::MOD_REPORTS\">\n"
       . "<input type=\"hidden\" name=\"view\"
value=\"$Reports::SAVE_REPORT\">\n"
       . "<input type=\"hidden\" name=\"case\" value=\"$Args::args{'case'}\">\n"
       . "<input type=\"hidden\" name=\"inv\" value=\"$Args::args{'inv'}\">\n";
```

```
  }
  #------------------------------
  # Case Info
  #------------------------------
  print "<strong><u>Case Information</u></strong>\n<ul>\n<li>"

      . ($SAVE ? "<a href=\"$CASEDIR/$MAP_HTML\" target=\"content\">" :
            "<a href=\"$REPORT_URL&view=$Reports::CASE_MAP\"
target=\"content\">")
      . "Case Map</a><p></li>\n<li>"

      . ($SAVE ? "<a href=\"$CASEDIR/$DETAILS_HTML\" target=\"content\">" :
            "<a href=\"$REPORT_URL&view=$Reports::CASE_DETAILS\"
target=\"content\">")
      . "Case Details</a><p></li>\n<li>";

  #determine default investigator to show details of
  my $inv = "unknown"; #default if no investigators have been added
  my @inv_list = Caseman::read_invest();
  if (scalar @inv_list > 0) {
    $inv = $inv_list[0];
  }
  print ""
      . ($SAVE ? "<a href=\"$INVDIR/$inv/$DETAILS_HTML\" target=\"content\">" :
            "<a href=\"$REPORT_URL&view=$Reports::INV_DETAILS&lu_inv=$inv\"
target=\"content\">")
      . "Investigators</a><p></li>\n"

      . "</ul>\n"

    #------------------------------
    # Case Audit
    #------------------------------
      . "<p><strong><u>Case Audit</u></strong>\n<ul>\n<li>"

      . ($SAVE ? "<a href=\"$AUDITDIR/$LOGS_HTML\" target=\"content\">" :
            "<a href=\"$REPORT_URL&view=$Reports::LOGS\"
target=\"content\">")
      . "Log Files</a><p></li>\n<li>"

      . ($SAVE ? "<a href=\"$AUDITDIR/$MD5S_HTML\" target=\"content\">" :
            "<a href=\"$REPORT_URL&view=$Reports::IMG_INTEG\"
target=\"content\">")
      . "Image Integrity</a><p></li>\n"
      . "</ul>\n"

    #------------------------------
    # Analysis Info
    #------------------------------
      . "<p><strong><u>Analysis Information</u></strong>\n<ul>\n<li>"

      . ($SAVE ? "<a href=\"$NOTESDIR/$INDEX_HTML\" target=\"content\">" :
            "<a href=\"$REPORT_URL&view=$Reports::NOTES_FRAME\"
target=\"content\">")
      . "Case Notes</a><p></li>\n<li>"

      . ($SAVE ? "<a href=\"$EVENTSDIR/$INDEX_HTML\" target=\"content\">" :
             "<a href=\"$REPORT_URL&view=$Reports::SEQ_FRAME\"
target=\"content\">")
      . "Sequencer Events</a><p></li>\n<li>"

      . ($SAVE ? "<a href=\"$KWORDDIR/$INDEX_HTML\" target=\"content\">" :
```

```
              "<a href=\"$REPORT_URL&view=$Reports::KWORD_FRAME\"
target=\"content\">")
      . "Keyword Searches</a><p></li>\n<li>"

      . ($SAVE ? "<a href=\"$FILESDIR/$FILECAT_HTML\" target=\"content\">" :
         "<a href=\"$REPORT_URL&view=$Reports::FILE_CAT_SUM\"
target=\"content\">")
      . "File Categories</a><p></li>\n<li>"

      . ($SAVE ? "<a href=\"$FILESDIR/$THUMBS_HTML\" target=\"content\">" :
         "<a href=\"$REPORT_URL&view=$Reports::THUMBS\" target=\"content\">")
      ."Picture Thumbnails</a><p></li><li>\n"

      . ($SAVE ? "<a href=\"$INDVREPDIR/$INDEX_HTML\" target=\"content\">" :
          "<a href=\"$REPORT_URL&view=$Reports::INDV_REP\"
target=\"content\">")
      . "Individual Reports</a><p></li>\n"

      . "</ul>\n";

  if (! $SAVE) {
    #----------------------------
    # Save Report
    #----------------------------
    print "<p> </p><center>Report Name:<br><input name=\"rname\"
type=\"text\" value=\"$Args::args{'case'}\" size=\"15\">\n"
       . "<br><br><input type=\"submit\" name=\"submit\" value=\"Save
Report\"></center>\n"
       . "</form></td></tr></table>\n";
  }

  Print::print_html_footer();
  return 0;
}

##############################################################################
# CASE INFORMATION SECTION

# Diagrams the hosts and disk images that have been added to the case
sub case_map {
  Print::print_report_html_header("Case Map for Case $Args::args{'case'}",
$SAVE);
  print "<p class=\"header01\" align=\"center\"><a name=\"top\"></a>Case
Map</p>\n";
  print_default();
  print "<p align=\"center\">The Case Map provides a hierarchal view of the
structure of this case based"
      . " <br>on the Hosts and Disk Images that have been added. For details
click"
      . " <br>on the Host and Volume links.</p><hr width=\"550\">";

  my %hosthash;
  my @hosts = get_hosts();

  if (scalar @hosts > 0) {
    print "<p align=\"center\"><strong>Hosts:</strong> | ";
    foreach my $h (sort @hosts) {
      #read image files
      $::host_dir = "$::case_dir" . "$h" . "/";
      $Args::args{'host'} = $Args::enc_args{'host'} = $h;
      Caseman::read_host_config();
```

```
      print "<a href=\"#$h\">$h</a> | ";
      my @imgs;
      my %imghash;

      if (scalar(keys %Caseman::vol2ftype) !=0) {
       foreach my $i (keys %Caseman::vol2cat) {
         if ($Caseman::vol2cat{$i} eq "image") {
           push @imgs, $i;
         }
       }
       foreach my $i (@imgs) {
         #Partitions
         my $partcount = scalar @{$Caseman::vol2part{$i}};
         my @parts;
         if ($partcount > 0) {
           foreach my $p (@{$Caseman::vol2part{$i}}) {
             push @parts, $p;
           }
         }
         #hash partition array to img
         @{ $imghash{$i} } = @parts;
       }
      }
      %{ $hosthash{$h} } = %imghash;
    }
    print "</p>\n";
  }
  else {
    print "<center>No Hosts have been added to this case. There is no"
       . " Case Map available.</center>\n";
  }
  print "<center><table border=\"0\" cellpadding=\"10\">\n";

  #print hosts
  foreach my $h (sort keys %hosthash) {
    $::host_dir = "$::case_dir" . "$h" . "/";
    $Args::args{'host'} = $Args::enc_args{'host'} = $h;
    Caseman::read_host_config();
    my $imgs = $hosthash{$h};

    print "<tr align=\"center\" valign=\"middle\"><td>\n"
       . "<a name=\"$h\"></a><table border=\"1\" cellspacing=\"0\"
cellpadding=\"10\" width=\"100%\">\n"
       . "<tr align=\"center\" valign=\"middle\"><td width=\"200\"
bgcolor=\"$::BACK_COLOR_TABLE\">\n"
       . "<strong>Host<br></strong>\n"
       . ($SAVE ? "<a href=\"../$HOSTDIR/$h/$DETAILS_HTML\">" :
                "<a href=\"$REPORT_URL&view=$Reports::HOST_DETAILS&host=$h\">")
       . "$h</a></td>\n"
       . "<td><table border=\"1\" cellspacing=\"0\" cellpadding=\"10\"
width=\"100%\">\n";
    if ( (scalar keys %{ $imgs }) > 0) {
      #print images
      foreach my $i (sort keys %{ $imgs }) {
       my $parts = $hosthash{$h}{$i};
       print "<tr align=\"center\" valign=\"middle\"><td width=\"200\"
bgcolor=\"$::BACK_COLOR_TABLE\">\n"
           . "<strong>Disk Image<br></strong>\n"
           . ($SAVE ? "<a href=\"../$HOSTDIR/$h/$IMGDIR/$i/$DETAILS_HTML\">" :
"<a href=\"$REPORT_URL&view=$Reports::IMG_DETAILS&host=$h&img=$i\">")
           . "$i</a><br>";
       if ($Caseman::vol2itype{$i} eq "split") {
```

```
        my $name = $Caseman::vol2splitname{$i};
        $name =~ s/,\s/<br>/g;
        print $name . "</td>\n";
      }
      else {
        print $Caseman::vol2sname{$i} . "</td>\n";
      }
      if ( (scalar @{ $parts }) > 0) {
        print "<td><table border=\"1\" cellspacing=\"0\" cellpadding=\"10\"
width=\"100%\">\n";

        #print partitions
        my $disk = 0;
        foreach my $p (@{ $parts }) {
          if ($Caseman::vol2cat{$p} eq "disk") {
            $disk = 1;
            print "<tr align=\"center\" valign=\"middle\"><td width=\"200\"
bgcolor=\"$::BACK_COLOR_TABLE\">\n"
              . "<strong>Raw<br></strong>\n"
              . ($SAVE ? "<a
href=\"../$HOSTDIR/$h/$IMGDIR/$i/$PARTDIR/$p/$DETAILS_HTML\">" :
                   "<a
href=\"$REPORT_URL&view=$Reports::PART_DETAILS&host=$h&img=$i&part=$p\">")
              . "$p</a><br>Disk<p><table border=\"1\" cellspacing=\"0\"
cellpadding=\"10\" width=\"100%\">\n";
          }
          else {
            print "<tr align=\"center\" valign=\"middle\"><td
bgcolor=\"$::BACK_COLOR_TABLE\">\n"
              . "<strong>Partition<br></strong>\n"
              . ($SAVE ? "<a
href=\"../$HOSTDIR/$h/$IMGDIR/$i/$PARTDIR/$p/$DETAILS_HTML\">" :
                  "<a
href=\"$REPORT_URL&view=$Reports::PART_DETAILS&host=$h&img=$i&part=$p\">")
              . "$p</a><br>$Caseman::vol2mnt{$p}</td></tr>\n";
          }
        }
        if ($disk) {
          print "</table></td></tr>\n";
        }
        print "</table></td>\n";
      }
      print "</tr>\n";
     }
   }
   else {
     print "<tr align=\"center\" valign=\"middle\"><td width=\"200\">\n"
       . "No Disk Images have been added to this host\n";
   }
   print "</table>\n</td>\n"
      . "</tr></table>"
      . "<p align=\"left\"><a href=\"#top\">^top</a></p></td></tr>\n";
 }
 print "</table></center>\n";
 Print::print_html_footer();
 return 0;
}

# Show the Case Details
sub case_details() {
    Print::print_report_html_header("Case Details for $Args::args{'case'}",
$SAVE);
```

```
    print "<p class=\"header01\" align=\"center\">Case Details</p>\n";
    print_default();

    Caseman::read_case_config();
    #print case name
    print "<table width=\"100%\">" .
         "<tr align=\"left\" valign=\"top\"><td colspan=\"2\">\n"
       . "<p class=\"header02\">Case \"$Args::args{'case'}\"</p></td>\n";

    #print location of case files
    print "<tr align=\"left\" valign=\"top\"><td nowrap><p
class=\"header03\">Case Directory: </p></td>\n" .
         "<td width=\"100%\">$::LOCKDIR/$Args::args{'case'}/</td></tr>\n";

    #print case description
    print "<tr align=\"left\" valign=\"top\"><td><p
class=\"header03\">Description: </p></td>\n" .
         "<td>$Caseman::cvals{'desc'}</td></tr>\n";

    #print case created
    print "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">Time
Created: </p></td>\n" .
         "<td>$Caseman::cvals{'created'}</td></tr>\n";

    #print investigators
    print "<tr align=\"left\" valign=\"top\"><td><p
class=\"header03\">Investigators: </p></td>\n<td>";

    # Display the valid investigators
    my @invs = Caseman::read_invest();
    if (scalar @invs < 1) {
       print "No investigators added to case. Default is \"unknown\"\n";
    }
    else {
      foreach my $i (@invs) {
       print ""
           . ($SAVE ? "<a href=\"../$INVDIR/$i/$DETAILS_HTML\">" :
              "<a href=\"$REPORT_URL&view=$Reports::INV_DETAILS&lu_inv=$i\">")
           . "$i</a><br>";
      }
    }
    print "<br></td></tr>\n";

    # print hosts
    print "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">Hosts:
</p></td>\n<td>";
    my @hosts = get_hosts();

    if (scalar @hosts >= 1) {
      foreach my $h (sort @hosts) {
       print "" . ( $SAVE ? "<a href=\"../$HOSTDIR/$h/$DETAILS_HTML\">" : "<a
href=\"$REPORT_URL&view=$Reports::HOST_DETAILS&host=$h\">")
          . "$h</a><br>\n\n";
      }
      print "<p> </p></td></tr>\n";
    }
    else {
      print "No hosts have been added to this case.</td></tr>\n";
    }

    # print case log
    my $caselog = $::case_dir . "case.log";
```

```
    print "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">Case
History: </p></td>\n<td>";
    if (-e "$caselog") {
      print "" . ($SAVE ? "<a href=\"$HIST_HTML\">" : "<a
href=\"$REPORT_URL&view=$Reports::PRINT_2HTML&show=caselog\">")
         . "$caselog</a>\n</td></tr>\n";
    }
    else {
      print "No log file found in $::case_dir.</td></tr>\n";
    }

    Print::print_html_footer();
    return 0;
}

# Show the Host Details
sub host_details {
  Print::print_report_html_header("Host Details for Case $Args::args{'case'}",
$SAVE);
  print "<table width=\"100%\" border=\"0\">\n"
      . "<tr align=\"left\" valign=\"top\"><td>\n"
     . "<p class=\"header01\" align=\"center\">Host Details</p>\n";
  print_default();

  my @tlinearray;
  my %md5hash = get_md5s();

  my $host = $Args::args{'host'};
  $::host_dir = "$::case_dir$host/";
  Caseman::read_host_config();

  #print host name
  print "<table width=\"100%\">\n"
      . "<tr align=\"left\" valign=\"top\"><td colspan=\"2\"><p
class=\"header02\">Host \"$host\"</td></tr>\n";

  #print host files directory
  print "<tr align=\"left\" valign=\"top\"><td nowrap><p
class=\"header03\">Host Directory: </p></td>\n" .
       "<td width=\"100%\">$::host_dir</td></tr>\n" .

  #print host description
       "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">Description:
</p></td>\n" .
       "<td>" . (($Caseman::host_desc ne "") ? $Caseman::host_desc : " ") .
       "</td></tr>\n" .

  #print host added time
       "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">Time Added:
</p></td>\n" .
       "<td>$Caseman::host_created</td></tr>\n" .

  #print investigator that added host
       "<tr align=\"left\" valign=\"top\"<td><p class=\"header03\">Investigator:
</p></td>\n<td>" .
       ($SAVE ? "<a href=\"../../$INVDIR/$Caseman::host_inv/$DETAILS_HTML\">" :
            "<a
href=\"$REPORT_URL&view=$Reports::INV_DETAILS&lu_inv=$Caseman::host_inv\">") .
       "$Caseman::host_inv" .

  #print timezone
```

```
      "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">Host
Timezone: </p></td>\n" .
      "<td>$Caseman::tz</td></tr>\n" .

  #print timeskew
      "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">Timeskew:
</p></td>\n" .
      "<td>$Caseman::ts</td></tr>\n";

  #print alert hash database
  print "<tr align=\"left\" valign=\"top\"><td nowrap><p
class=\"header03\">Alert Hash Database: </p></td>\n<td>";
  if ($Caseman::alert_db ne "") {
    if (-e "$Caseman::alert_db") {
      print "$Caseman::alert_db</td></tr>\n";
    }
  }

  #print exclude hash database
  print "<tr align=\"left\" valign=\"top\"><td nowrap><p
class=\"header03\">Exclude Hash Database: </p></td>\n<td>";
  if ($Caseman::exclude_db ne "") {
    if (-e "$Caseman::exclude_db") {
      print "$Caseman::exclude_db</a></td></tr>\n";
    }
  }

  #print image files
  print "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">Images:
</p></td>\n<td>";
  if (scalar(keys %Caseman::vol2ftype) !=0) {
    my @imgarray;

    foreach my $i (keys %Caseman::vol2cat) {
      if ($Caseman::vol2cat{$i} eq "image") {
       push @imgarray, $i;
      }
      elsif (($Caseman::vol2ftype{$i} eq "body") || ($Caseman::vol2ftype{$i} eq
"timeline")) {
       push @tlinearray, $i;
      }
    }
    my @imgsort = sort { (lc($a) cmp lc($b)) } @imgarray;
    foreach my $i (@imgsort) {
      print "<table border=\"0\">\n<tr align=\"left\" valign=\"top\"><td>"
         . ($SAVE ? "<a href=\"$IMGDIR/$i/$DETAILS_HTML\">" : "<a
href=\"$REPORT_URL&view=$Reports::IMG_DETAILS&host=$host&img=$i\">")
         . "$i</a>     </td>";
      if ($Caseman::vol2itype{$i} eq "split") {
       print "<td>$Caseman::vol2splitname{$i}<br></td></tr></table>\n";
      }
      else {
       print "<td>$Caseman::vol2sname{$i}<br>\n</td></tr></table>";
      }
    }
    print "<p> </p></td></tr>\n";
  }

  else {
    print "No image files have been added to this host.</td></tr>\n";
  }
```

```
  #print host log
  my $hostlog = $::host_dir . "$::LOGDIR/host.log";
  print "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">Host
History: </p></td>\n<td>";
  if (-e "$hostlog") {
    print "" . ($SAVE ? "<a href=\"$HIST_HTML\">" : "<a
href=\"$REPORT_URL&view=$Reports::PRINT_2HTML&host=$host&show=hostlog\">")
       . "$hostlog</a><p> </p>\n</td></tr>\n";
  }
  else {
    print "No log file found in $::host_dir/logs.</td></tr>\n";
  }

  # timeline files
  print "<tr align=\"left\" valign=\"top\"<td><p class=\"header03\">Timeline
Files:</p></td>\n";
  if (scalar(@tlinearray) != 0) {
    my @tlinesort = sort { (lc($a) cmp lc($b)) } @tlinearray;
      print "<td colspan=\"2\"><table border=\"1\" cellspacing=\"0\" "
         . "cellpadding=\"8\">" .
           "<tr align=\"center\" valign=\"top\" bgcolor=\"$BGCOLOR\"><td><b>File
Name</b></td>\n<td><b>Type</b></td>\n<td><b>Created</b></td>\n" .
           "<td><b>Investigator</b></td>";

    my $row = 0;
    my $bgcolor;
    foreach my $i (@tlinesort) {
      if ($row++ % 2 == 0) {
       $bgcolor = $::BACK_COLOR_TABLE;
      }
      else {
       $bgcolor = $::BACK_COLOR;
      }
      print
           "<tr align=\"center\" valign=\"top\"
bgcolor=\"$bgcolor\"><td>$Caseman::vol2sname{$i}</td>\n" .
           "<td>" . (($Caseman::vol2ftype{$i} eq "body") ? "body" : "timeline")
.
           "<td>" . (($Caseman::vol2tadded{$i} ne "") ? $Caseman::vol2tadded{$i}
: "unknown") .
           "<td>" . (($Caseman::vol2iadded{$i} ne "") ? $Caseman::vol2iadded{$i}
: "unknown") .
           "</td></tr>\n";
      if ($md5hash{$i}) {
       print "<tr align=\"right\" bgcolor=\"$bgcolor\"><td colspan=\"4\"> 
  "
           . "<strong>md5:</strong> $md5hash{$i}</td></tr>\n";
      }
    }
    print "</table></td>\n";
  }

  print "</tr></table>\n";
  Print::print_html_footer();
  return 0;
}

# Show the Image Details
sub img_details {
  Print::print_report_html_header("Image Details for Case $Args::args{'case'}",
$SAVE);
  print "<table width=\"100%\" border=\"0\">\n<tr><td>\n"
```

```
      . "<p class=\"header01\" align=\"center\">Image Details</p>\n";
  print_default();

  my %md5hash = get_md5s();

  my $host = $Args::args{'host'};
  my $img = $Args::args{'img'};
  $::host_dir = "$::case_dir" . "$host" . "/";
  Caseman::read_host_config();

  #Image name
  print "<table width=\"100%\" border=\"0\">"
      . "<tr align=\"left\" valign=\"top\"><td colspan=\"2\"><p
class=\"header02\">"
      . "Image \"$Caseman::vol2sname{$img}\" on Host \"$host\"
</p></td></tr>\n";

  #Path
  my $img_path = $Caseman::vol2path{$img};
  $img_path =~ s/'//g;

  print "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">"
      . "Image Path: </td>\n<td width=\"100%\">$img_path</p></td></tr>\n"

  #Time added to host
      . "<tr align=\"left\" valign=\"top\"><td nowrap><p
class=\"header03\">Time Added: </p></td>\n"
      . "<td>$Caseman::vol2tadded{$img}</p></td></tr>\n"

  #Investigator
      . "<tr align=\"left\" valign=\"top\"><td><p
class=\"header03\">Investigator: </p></td>\n"
      . "<td>$Caseman::vol2iadded{$img}</p></td></tr>\n"

  #Volume ID
      . "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">Volume ID:
</p></td>\n"
      . "<td>$img</td></tr>\n";

  #MD5
  print "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">MD5:
</p></td>\n";
  if (exists $md5hash{$img}) {
    print "<td>$md5hash{$img}</td></tr>\n";
  }
  else {
    print "<td>$img Autopsy has not been used to acquire an md5
value.</td></tr>\n";
  }

  #Partitions
  my $partcount = scalar @{$Caseman::vol2part{$img}};
  print "<tr align=\"left\" valign=\"top\"><td><p
class=\"header03\">Partitions: </p></td><td>\n";

  if ($partcount > 0) {
    print "<table>";
    foreach my $p (@{$Caseman::vol2part{$img}}) {
      #volume id
      print "<tr><td align=\"left\">"
         . ($SAVE ? "<a href=\"$PARTDIR/$p/$DETAILS_HTML\">" :
```

```
                "<a
href=\"$REPORT_URL&view=$Reports::PART_DETAILS&host=$host&img=$img&part=$p\">")
        . "$p</a></td>\n"
      #autopsy display name
         . "<td align=\"left\">    $Caseman::vol2sname{$p}</td>\n" .
      #mount name
          "<td align=\"left\">    $Caseman::vol2mnt{$p}</td>\n" .
      #filesystem type
          "<td align=\"left\">   
$Caseman::vol2ftype{$p}</td></tr>\n";
    }
    print "</table>\n";
  }
  else {
    print "This is a raw image with no partitions.\n";
  }

  print "</table>\n</tr><td></table>";
  Print::print_html_footer();
  return 0;
}

# Show the Partition Details
sub part_details {
  Print::print_report_html_header("Partition Details for Case
$Args::args{'case'}", $SAVE);
  print "<table width=\"100%\" border=\"0\">\n<tr><td>\n"
      . "<p class=\"header01\" align=\"center\">Partition Details</p>\n";
  print_default();

  my $host = $Args::args{'host'};
  my $img = $Args::args{'img'};
  my $part = $Args::args{'part'};
  $::host_dir = "$::case_dir" . "$host" . "/";
  Caseman::read_host_config();

  my %md5hash = get_md5s();

  #Partition name
  print "<table width=\"100%\" border=\"0\">"
      . "<tr align=\"left\" valign=\"top\"><td colspan=\"2\"><p
class=\"header02\">"
      . "Partition \"$Caseman::vol2sname{$part}\" on host \"$host\"
</p></td></tr>\n";

  #Mount name
  print "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">Mount
Name: </p></td>\n"
      . "<td width=\"100%\">$Caseman::vol2mnt{$part}</td></tr>\n";

  #File System
  print "<tr align=\"left\" valign=\"top\"><td nowrap><p
class=\"header03\">File System: </p></td>\n"
      . "<td>$Caseman::vol2ftype{$part}";
  #File system details
  print "    "
      . ($SAVE ? "<a href=\"fsdetails-$part.html\">" :
         "<a
href=\"$REPORT_URL&view=$Reports::FS_DETAILS&host=$host&vol=$part\">")
      . "details</a>";
  print "</td></tr>\n";
```

```
  #Volume ID
  print "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">Volume ID:
</p></td>\n"
      . "<td>$part</td></tr>\n";

  #Start and End
  print "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">Sectors:
</p></td>\n";
  if (($Caseman::vol2end{$part} eq "0") || ($Caseman::vol2cat{$part} eq
"disk")) {
    print "<td>This partition represents the entire image.</td></tr>\n";
  }
  else {
    print "<td>$Caseman::vol2start{$part} -
$Caseman::vol2end{$part}</td></tr>\n";
  }

  print "</table>\n";

  #Extracted Files
  print "<table><tr align=\"left\" valign=\"top\"><td colspan=\"2\"><br><p
class=\"header03\">"
      . "<u>Extracted Files</u></p></td></tr>\n";

  # Strings File
  if (exists $Caseman::vol2str{$part}) {
    my $strvol = $Caseman::vol2str{$part};
    print "<tr align=\"left\" valign=\"top\"><td><table border=\"1\" "
       . "cellspacing=\"0\" cellpadding=\"3\">"
       . "<tr align=\"left\" valign=\"top\" bgcolor=\"$BGCOLOR\">\n"
       . "<td colspan=\"4\"><b>ASCII Strings:</b></td></tr>"
       . "<tr align=\"center\"
bgcolor=\"$::BACK_COLOR_TABLE\"><td><b>VolID</b></td>"
       .
"<td><b>Name</b></td><td><b>Created</b></td><td><b>Investigator</b></td></tr>"
       . "<tr><td>$strvol</td>"
       . "<td>$Caseman::vol2sname{$strvol}</td>\n"
       . "<td>"
       . ($Caseman::vol2tadded{$strvol} ? "$Caseman::vol2tadded{$strvol}" :
"unknown")
       . "</td>\n<td>"
       . ($Caseman::vol2iadded{$strvol} ? "$Caseman::vol2iadded{$strvol}" :
"unknown")
       . "</td></tr>\n";
    if ($md5hash{$strvol}) {
      print "<tr><td align=\"right\" colspan=\"4\">     <b>MD5:</b>
$md5hash{$strvol}</td></tr>\n";
    }
    print "</table><p> </p></td></tr>\n";
  }

  # Unicode Strings File
  if (exists $Caseman::vol2uni{$part}) {
    my $univol = $Caseman::vol2uni{$part};
    print "<tr align=\"left\" valign=\"top\"><td><table border=\"1\" "
       . "celspacing=\"0\" cellpadding=\"3\">"
       . "<tr align=\"left\" valign=\"top\" bgcolor=\"$BGCOLOR\">\n"
       . "<td colspan=\"4\"><b>Unicode Strings:</b></td></tr>"
       . "<tr align=\"center\"
bgcolor=\"$::BACK_COLOR_TABLE\"><td><b>VolID</b></td><td>"
       .
"<b>Name</b></td><td><b>Created</b></td><td><b>Investigator</b></td></tr>"
```

```
       . "<tr><td>$univol</td>"
       . "<td>$Caseman::vol2sname{$univol}</td>\n"
       . "<td>"
       . ($Caseman::vol2tadded{$univol} ? "$Caseman::vol2tadded{$univol}" :
"unknown")
       . "</td>\n<td>"
       . ($Caseman::vol2iadded{$univol} ? "$Caseman::vol2iadded{$univol}" :
"unknown")
       . "</td></tr>\n";
    if ($md5hash{$univol}) {
      print "<tr><td align=\"right\" colspan=\"4\">     <b>MD5:</b>
$md5hash{$univol}</td></tr>\n";
    }
    print "</table><p> </p</td></tr>\n";
  }

  # dls file
  if (exists $Caseman::vol2dls{$part}) {
    my $dlsvol = $Caseman::vol2dls{$part};
    print "<tr align=\"left\" valign=\"top\"><td><table border=\"1\" "
       . "cellspacing=\"0\" cellpadding=\"3\">"
       . "<tr align=\"left\" valign=\"top\" bgcolor=\"$BGCOLOR\">\n"
       . "<td colspan=\"4\"><b>Unallocated
$Fs::addr_unit{$Caseman::vol2ftype{$part}}s:</b></td>"
       . "<tr align=\"center\"
bgcolor=\"$::BACK_COLOR_TABLE\"><td><b>VolID</b></td><td>"
       .
"<b>Name</b></td><td><b>Created</b></td><td><b>Investigator</b></td></tr>"
       . "<tr><td>$dlsvol</td>"
       . "<td>$Caseman::vol2sname{$dlsvol}</td>\n"
       . "<td>"
       . ($Caseman::vol2tadded{$dlsvol} ? "$Caseman::vol2tadded{$dlsvol}" :
"unknown")
       . "</td>\n<td>"
       . ($Caseman::vol2iadded{$dlsvol} ? "$Caseman::vol2iadded{$dlsvol}" :
"unknown")
       . "</td></tr>\n";
    if ($md5hash{$dlsvol}) {
      print "<tr><td align=\"right\" colspan=\"4\">     <b>MD5:</b>
$md5hash{$dlsvol}</td></tr>\n";
    }
    print "</table><p> </p</td></tr>\n";

    # strings of dls
    if (exists $Caseman::vol2str{$Caseman::vol2dls{$part}}) {
      my $dlsstrvol = $Caseman::vol2str{$dlsvol};
      print "<tr align=\"left\" valign=\"top\"><td><table border=\"1\" "
         . "cellspacing=\"0\" cellpadding=\"3\"><tr align=\"center\">"
         . "<tr align=\"left\" valign=\"top\" bgcolor=\"$BGCOLOR\">\n"
         . "<td colspan=\"4\"><b>ASCII Strings of Unallocated:</b></td>"
         . "<tr align=\"center\"
bgcolor=\"$::BACK_COLOR_TABLE\"><td><b>VolID</b></td><td>"
         .
"<b>Name</b></td><td><b>Created</b></td><td><b>Investigator</b></td></tr>"
         . "<tr><td>$dlsstrvol</td>"
         . "<td>$Caseman::vol2sname{$dlsstrvol}</td>\n"
         . "<td>"
         . ($Caseman::vol2tadded{$dlsstrvol} ?
"$Caseman::vol2tadded{$dlsstrvol}" : "unknown")
         . "</td>\n<td>"
         . ($Caseman::vol2iadded{$dlsstrvol} ?
"$Caseman::vol2iadded{$dlsstrvol}" : "unknown")
```

```
         . "</td></tr>\n";
       if ($md5hash{$dlsstrvol}) {
        print "<tr><td align=\"right\" colspan=\"4\">     <b>MD5:</b>
$md5hash{$dlsstrvol}</td></tr>\n";
       }
       print "</table><p> </p</td></tr>\n";
     }

     # unistrings of dls
     if (exists $Caseman::vol2uni{$Caseman::vol2dls{$part}}) {
       my $dlsunivol = $Caseman::vol2uni{$dlsvol};
       print "<tr align=\"left\" valign=\"top\"><td><table border=\"1\" "
          . "cellspacing=\"0\" cellpadding=\"3\">"
          . "<tr align=\"left\" valign=\"top\" bgcolor=\"$BGCOLOR\">\n"
          . "<td colspan=\"4\"><b>Unicode Strings of Unallocated:</b></td>"
          . "<tr align=\"center\"
bgcolor=\"$::BACK_COLOR_TABLE\"><td><b>VolID</b></td><td>"
          .
"<b>Name</b></td><td><b>Created</b></td><td><b>Investigator</b></td></tr>"
          . "<tr><td>$dlsunivol</td>"
          . "<td>$Caseman::vol2sname{$dlsunivol}</td>\n"
          . "<td>"
          . ($Caseman::vol2tadded{$dlsunivol} ?
"$Caseman::vol2tadded{$dlsunivol}" : "unknown")
          . "</td>\n<td>"
          . ($Caseman::vol2iadded{$dlsunivol} ?
"$Caseman::vol2iadded{$dlsunivol}" : "unknown")
          . "</td></tr>\n";
       if ($md5hash{$dlsunivol}) {
        print "<tr><td align=\"right\" colspan=\"4\">     <b>MD5:</b>
$md5hash{$dlsunivol}</td></tr>\n";
       }
       print "</table></td></tr>\n";
     }
   }
   print "</table>\n</td></tr></table>\n";
   Print::print_html_footer();
   return 0;
}

# Show the File System Details
sub fs_details {
   my $host = $Args::args{'host'};
   $::host_dir = "$::case_dir" . "$Args::args{'host'}" . "/";
   Caseman::read_host_config();

   my $vol = Args::get_vol('vol');
   my $img     = $Caseman::vol2path{$vol};
   my $imgtype = $Caseman::vol2itype{$vol};
   my $offset  = $Caseman::vol2start{$vol};

   if ($Caseman::vol2cat{$vol} ne "disk") {
     my $ftype   = $Caseman::vol2ftype{$vol};
     Print::print_report_html_header("File System Details", $SAVE);
     print "<p class=\"header01\" align=\"center\">File System Details</p>\n";
     print_default();

     my $fat = 0;
     my $inv = Args::get_inv();

     Exec::exec_pipe(*OUT,"'$::TSKDIR/fsstat' -f $ftype -o $offset -i $imgtype
$img");
```

```
  while ($_ = Exec::read_pipe_line(*OUT)) {

     if (/\-\-\-\-\-\-\-\-\-\-\-/) {

         # Ignore these and print them ahead of the headers
     }

     # need the space to prevent NTFS STD_INFORMATION from triggering it
     elsif (/ INFORMATION/) {
         print "<b>$_</b><p>\n";
     }
     elsif (($ftype =~ /fat/) && ($_ =~ /FAT CONTENTS/)) {
         print "<hr><b>$_</b><p>\n";

         # Set the flag if we reach the FAT
         $fat = 1;
     }

     # Special case for FAT
     # We will be giving hyperlinks in the FAT table dump
     elsif ($fat == 1) {

         # Ignore the divider
         if (/\-\-\-\-\-\-\-\-\-\-\-/) {
            print "$_<br>";
            next;
         }

         if (/^((\d+)\-\d+\s+\((\d+)\)) \-\> ([\w]+)$/) {
            my $full = $1;
            my $blk  = $2;
            my $len  = $3;
            my $next = $4;

            # Print the tag so that other FAT entries can link to it
            print "<a name=\"$blk\">\n";

            print "$full -> ";

            if ($next eq 'EOF') {
                print "EOF<br>\n";
            }
            else {
                print "<a href=\"#$next\">$next</a><br>\n";
            }
         }
         else {
            $fat = 0;
            print "$_<br>";
         }
     }
     else {
         print "$_<br>";
     }
  }
  close(OUT);
}

elsif ($Caseman::vol2cat{$vol} eq "disk") {
  my $dtype = $Caseman::vol2dtype{$vol};

  Print::print_report_html_header("Disk Details Details", $SAVE);
```

```
    print "<p class=\"header01\" align=\"center\">Disk Details</p>\n";
    print_default();
    print "<b>PARTITION INFORMATION</b><p>\n";

    Exec::exec_pipe(*OUT,"'$::TSKDIR/mmls' -o $offset -i $imgtype -t $dtype -r
$img");
    while ($_ = Exec::read_pipe_line(*OUT)) {
       print "<tt>$_</tt><br>\n";
    }
    close(OUT);
  }

  Print::print_html_footer();
  return 0;
}

# Show the Investigator Details
sub inv_details {
  # begin printing html frames
  Print::print_report_html_header("Investigator Details for Case
$Args::args{'case'}", $SAVE);
  print "<p class=\"header01\" align=\"center\">Investigator Details</p>\n";
  print_default();

  my @invest = Caseman::read_invest();
  if ( inv_unknown() ) {
    push @invest, "unknown"; #default investigator
  }
  # Get investigator to be shown
  my $inv = "";
  if (exists $Args::args{'lu_inv'}) {
    $inv = $Args::args{'lu_inv'};
  }
  else {
    Print::print_err("Error: No Investigator was specified");
  }

  #list of case investigators
  print "<p align=\"center\">\n<strong>Case Investigators:</strong> | ";
  if (scalar @invest > 0) {
    foreach my $i (@invest) {
      print ""
         . ($SAVE ? "<a href=\"../$i/$DETAILS_HTML\">" :
                    "<a
href=\"$REPORT_URL&view=$Reports::INV_DETAILS&lu_inv=$i\">")
         . "$i </a>| \n";
    }
  }

  print "</p>\n"
      . "<table width=\"100%\">\n"
      . "<tr align=\"left\" valign=\"top\"><td colspan=\"2\">"
      . "<p class=\"header02\">Investigator \"$inv\"</p>\n";

  #Read investigator info file if it exists
  if (-e "$::case_dir/$inv.aut") {
    open IINFO, "$::case_dir/$inv.aut" or die "Can't open $::case_dir/$inv.aut
: $!";
    while (<IINFO>) {
       if (/^comp\s(.*)$/) {
         print "</td></tr>\n<tr align=\"left\" valign=\"top\"><td><p
class=\"header03\">Company/Agency: </td>\n"
```

```
            . "<td width=\"100%\">$1";
      }
      elsif (/^title\s(.*)$/) {
        print "</td></tr>\n<tr align=\"left\" valign=\"top\"><td><p
class=\"header03\">Title: </td>\n"
            . "<td width=\"100%\">$1";
      }
      elsif (/^addr\s(.*)$/) {
        print "</td></tr>\n<tr align=\"left\" valign=\"top\"><td><p
class=\"header03\">Address: </td>\n"
            . "<td width=\"100%\">$1";
      }
      elsif (/^phone\s(.*)$/) {
        print "</td></tr>\n<tr align=\"left\" valign=\"top\"><td><p
class=\"header03\">Phone: </td>\n"
            . "<td width=\"100%\">$1";
      }
      elsif (/^email\s(.*)$/) {
        print "</td></tr>\n<tr align=\"left\" valign=\"top\"><td><p
class=\"header03\">Email: </td>\n"
            . "<td width=\"100%\">$1";
      }
      elsif (/^comm\s(.*)$/) {
        print "</td></tr>\n<tr align=\"left\" valign=\"top\"><td><p
class=\"header03\">Comment: </td>\n"
            . "<td width=\"100%\">$1";
      }
      else {
        print "<br>$1" if (/^(.*)$/);
      }
    }
    print "</td></tr>";
    }

    else {
      print "</td></tr><tr><td colspan=\"2\">"
         . "This investigator does not have any personal information
saved.</td></tr>\n";
    }

    # Notes
    print "<tr><td><p> </p><tr align=\"left\" valign=\"top\">\n"
       . "<td><p class=\"header03\">Notes: </td><td width=\"100%\">\n";

    #Scan Host directories for investigator notes
    my @hosts = get_hosts();
    foreach my $h (sort @hosts) {
      my $notesfile = "$::case_dir$h/$::LOGDIR/$inv.notes";
      if (-e "$notesfile") {
       print "<strong>Host $h</strong><br>"
           . ($SAVE ? "<a href=\"$NOTES_HTML#$h\" target=\"content\">" :
                    "<a
href=\"$REPORT_URL&view=$Reports::INV_NOTES&luinv=$inv#$h\"
target=\"content\">")
           . "$notesfile</a><p></p>\n";
      }
    }
    print "<p> </p></td></tr>\n";

    # History
    print "<tr align=\"left\" valign=\"top\"><td><p class=\"header03\">History:
</p></td>\n<td>\n";
```

```
    foreach my $h (sort @hosts) {
      my $invlog = "$::case_dir$h/$::LOGDIR/$inv.log";
      if (-e "$invlog") {
       print "<strong>Host $h</strong><br>"
           . ($SAVE ? "<a href=\"$h$HIST_HTML\" target=\"content\">" :
              "<a
href=\"$REPORT_URL&view=$Reports::PRINT_2HTML&host=$h&luinv=$inv&show=invlog\"
target=\"content\">")
           . "$invlog</a><p></p>\n";
      }

    }
    print "<p> </p></td></tr>\n";

    # Command History
    print "<tr align=\"left\" valign=\"top\"><td nowrap><p
class=\"header03\">Command Log: </p></td>\n<td>\n";
    foreach my $h (sort @hosts) {
      my $invcomm = "$::case_dir$h/$::LOGDIR/$inv.exec.log";
      if (-e "$invcomm") {
       print "<strong>Host $h</strong><br>"
           . ($SAVE ? "<a href=\"$h$COM_HTML\" target=\"content\">" :
              "<a
href=\"$REPORT_URL&view=$Reports::PRINT_2HTML&host=$h&luinv=$inv&show=invcomm\"
target=\"content\">")
           . "$invcomm</a><p></p>\n";
      }
    }
  print "<p> </p></td></tr></table>\n";

  Print::print_html_footer();
  return;
}

# END CASE INFORMATION SECTION
################################################################################

################################################################################
# CASE AUDIT SECTION

# Show case, host, investigator, and command logs
sub logs {
  Print::print_report_html_header("Log Files", $SAVE);
  print "<p class=\"header01\" align=\"center\">Log Files</p>\n";
  print_default();

  #Case Log
  my $fname = $::case_dir . "case.log";
  print "<table border=\"0\"><tr align=\"left\" valign=\"top\">"
      . "<td nowrap><p class=\"header02\">Case History: </p></td>\n"
      . "<td width=\"100%\">";
  if (-e "$fname") {
    print ""
       . ($SAVE ? "<a href=\"../$CASEDIR/$HIST_HTML\">" :
                "<a
href=\"$REPORT_URL&view=$Reports::PRINT_2HTML&show=caselog\">")
       . "$fname</a>\n";
  }
  else {
    print "No log file found in $::case_dir.\n";
  }
  print "</td></tr>\n";
```

```
  my @hosts = get_hosts();
  my @inv = Caseman::read_invest();

  print "<tr align=\"left\" valign=\"top\">\n<td nowrap>"
      . "<p class=\"header02\">Host History: </p></td>\n"
      . "<td width=\"100%\"><table border=\"0\" cellspacing=\"0\" "
      . "cellpadding=\"3\">\n";
  foreach my $h (sort @hosts) {
    print "<tr align=\"center\" bgcolor=\"$BGCOLOR\">\n"
       . "<td><strong>$h</strong></td></tr>\n";

    $Args::args{'host'} = $Args::enc_args{'host'} = $h;
    $::host_dir = "$::case_dir$h/";
    $fname = $::host_dir . "$::LOGDIR/host.log";
    if (-e "$fname") {
      #Host Logs
      print "<tr align=\"left\"><td>"
         . ($SAVE ? "<a href=\"../$HOSTDIR/$h/$HIST_HTML\">"
                  : "<a
href=\"$REPORT_URL&view=$Reports::PRINT_2HTML&host=$h&show=hostlog\">")
         . "$fname</a></td></tr>\n";
    }
    else {
      print "<tr align=\"left\"><td>"
         . "No log file found in $::case_dir$h.</td></tr>\n";
    }
    #Investigator Logs
    print "<tr align=\"left\"><td>"
       . "<strong><p></p>Investigator History: </strong></td></tr>\n";
    foreach my $i (@inv) {
      $fname = "$::case_dir$h/$::LOGDIR/$i.log";
      if (-e "$fname") {
       print "<tr align=\"left\"><td>"
           . ($SAVE ? "<a href=\"../$INVDIR/$i/$h$HIST_HTML\">" :
              "<a
href=\"$REPORT_URL&view=$Reports::PRINT_2HTML&host=$h&luinv=$i&show=invlog\">")
           . "$fname</a></td></tr>\n";
      }
    }
    #check for "unknown" investigator
    $fname = "$::case_dir$h/$::LOGDIR/unknown.log";
    if (-e $fname) {
      print "<tr align=\"left\"><td>"
         . ($SAVE ? "<a href=\"../$INVDIR/unknown/$h$HIST_HTML\">" :
                "<a
href=\"$REPORT_URL&view=$Reports::PRINT_2HTML&host=$h&luinv=unknown&show=invlog
\">")
         . "$fname</a></td></tr>\n";
    }

    #Investigator commands
    print "<tr align=\"left\">\n"
       . "<td><strong><p></p>Investigator Commands: </strong></td></tr>\n";
    foreach my $i (@inv) {
      $fname = "$::case_dir$h/$::LOGDIR/$i.exec.log";
      if (-e $fname) {
       print "<tr align=\"left\"><td>"
           . ($SAVE ? "<a href=\"../$INVDIR/$i/$h$COM_HTML\">" :
                      "<a
href=\"$REPORT_URL&view=$Reports::PRINT_2HTML&host=$h&luinv=$i&show=invcomm\">"
)
```

```
            . "$fname</a></td></tr>\n";
        }
      }
      #check for "unknown" investigator
      $fname = "$::case_dir$h/$::LOGDIR/unknown.exec.log";
      if (-e $fname) {
        print "<tr align=\"left\"><td>"
           . ($SAVE ? "<a href=\"../$INVDIR/unknown/$h$COM_HTML\">" :
                "<a
href=\"$REPORT_URL&view=$Reports::PRINT_2HTML&host=$h&luinv=unknown&show=invcom
m\">")
           . "$fname</a></td></tr>\n";
      }

      print "<tr><td><p> </p></td></tr>\n";
    }
    print "</td></tr></table></table></td></tr></table>\n";
    Print::print_html_footer();
    return 0;
}

# Show MD5 list for disk images and volumes
sub img_integ {
    Print::print_report_html_header("Image Integrity", $SAVE);
    print "<p class=\"header01\" align=\"center\">Image Integrity</p>\n";
    print_default();

    my @hosts = get_hosts();
    if (scalar @hosts < 1) {
      print "<center>No Hosts have been added to this case</center>";
    }
    print "<center><table border=\"0\">\n";
    foreach my $h (sort @hosts) {
      $::host_dir = "$::case_dir$h/";
      $Args::args{'host'} = $Args::enc_args{'host'} = $h;
      Caseman::read_host_config();

      my %md5s = get_md5s();
      my @imgs;
      my @dls;
      my @strings;
      my @body;
      my @tl;
      if (scalar(keys %Caseman::vol2ftype) !=0) {
        foreach my $i (keys %Caseman::vol2cat) {
         if ($Caseman::vol2cat{$i} eq "image") {
           push @imgs, $i;
         }
         elsif ($Caseman::vol2ftype{$i} eq "dls") {
           push @dls, $i;
         }
         elsif ($Caseman::vol2ftype{$i} eq "strings") {
           push @strings, $i;
         }
         elsif ($Caseman::vol2ftype{$i} eq "body") {
           push @body, $i;
         }
         elsif ($Caseman::vol2ftype{$i} eq "timeline") {
           push @tl, $i;
         }
        }
      }
```

```
    #host name
    print "<tr><td colspan=2><p> </p></td></tr>\n"
       . "<tr align=\"center\" bgcolor=\"$BGCOLOR\"><td colspan=\"3\"><p
class=\"header02\">"
       . "$h</p></td></tr>"
       #File System Images
       . "<tr align=\"left\" bgcolor=\"$::BACK_COLOR_TABLE\">"
       . "<td colspan=2><p class=\"header03\">File System
Images</p></td></tr>\n";
    foreach my $i (@imgs) {
      print "<tr align=\"left\"><td><p class=\"header04\"><strong>"
         . "$Caseman::vol2sname{$i}    </strong></p></td>\n<td>";
      if (exists $md5s{$i}) {
       print "$md5s{$i}</td></tr>\n";
      }
      else {
       print "Not computed</td></tr>\n";
      }
    }
    #Unallocated Data Files
    print "<tr><td colspan=2><p> </p></td></tr>\n"
       . "<tr align=\"left\" bgcolor=\"$::BACK_COLOR_TABLE\">"
       . "<td colspan=2><p class=\"header03\">Unallocated Data
Files</p></td></tr>\n";
    foreach my $d (@dls) {
      print "<tr align=\"left\"><td><p class=\"header04\"><strong>"
         . "$Caseman::vol2sname{$d}    </strong></p></td>\n<td>";
      if (exists $md5s{$d}) {
       print "$md5s{$d}</td></tr>\n";
      }
      else {
       print "Not computed</td></tr>\n";
      }
    }
    #String Files
    print "<tr><td colspan=2><p> </p></td></tr>\n"
       . "<tr align=\"left\" bgcolor=\"$::BACK_COLOR_TABLE\">"
       . "<td colspan=2><p class=\"header03\">Strings Files</p></td></tr>\n";
    foreach my $s (@strings) {
      print "<tr align=\"left\"><td><p class=\"header04\"><strong>"
         . "$Caseman::vol2sname{$s}    </strong></p></td>\n<td>";
      if (exists $md5s{$s}) {
       print "$md5s{$s}</td></tr>\n";
      }
      else {
       print "Not computed</td></tr>\n";
      }
    }
    #Timeline Data Files
    print "<tr><td colspan=2><p> </p></td></tr>\n"
       . "<tr align=\"left\" bgcolor=\"$::BACK_COLOR_TABLE\">"
       . "<td colspan=2><p class=\"header03\">Timeline Data
Files</p></td></tr>\n";
    foreach my $b (@body) {
      print "<tr align=\"left\"><td><p class=\"header04\"><strong>"
         . "$Caseman::vol2sname{$b}    </strong></p></td>\n<td>";
      if (exists $md5s{$b}) {
       print "$md5s{$b}</td></tr>\n";
      }
      else {
       print "Not computed</td></tr>\n";
      }
```

```
    }
    #Timeline Files
    print "<tr><td colspan=2><p> </p></td></tr>\n"
       . "<tr align=\"left\" bgcolor=\"$::BACK_COLOR_TABLE\">"
       . "<td colspan=2><p class=\"header03\">Timeline Files</p></td></tr>\n";
    foreach my $t (@tl) {
      print "<tr align=\"left\"><td><p class=\"header04\"><strong>"
         . "$Caseman::vol2sname{$t}    </strong></p></td>\n<td>";
      if (exists $md5s{$t}) {
       print "$md5s{$t}</td></tr>\n";
      }
      else {
       print "Not computed</td></tr>\n";
      }
    }
    print "<tr><td><p> </p></td></tr>\n"
  }
  print "</table></center>\n";
  Print::print_html_footer();
  return 0;
}

# END CASE AUDIT SECTION
################################################################################

################################################################################
# ANALYSIS INFORMATION SECTION

# Generate html frames page for displaying case notes
sub notes_frame {
  # begin printing html frames
  Print::print_html_header_frameset("Bookmarks for Case $Args::args{'case'}",
$SAVE);
  print "<frameset rows=\"250,100%\" framespacing=\"0\" border=\"0\">\n";

  if ($SAVE) {
    print "<frame src=\"$BMARKS_HTML\">\n"
       . "<frame src=\"$NOTES_HTML\" name=\"notes\">\n</frameset>\n";
  }
  else {
    print "<frame src=\"$REPORT_URL&view=$Reports::NOTES_BMARKS\">\n"
       . "<frame src=\"$REPORT_URL&view=$Reports::NOTES_ALL\" "
       . "name=\"notes\">\n</frameset>\n";
  }
  print <<EOF;
<NOFRAMES>
  <center>
    Autopsy requires a browser that supports frames.
  </center>
</NOFRAMES>
EOF

  Print::print_html_footer_frameset();
  return;
}

# Display in frame list of note bookmarks
sub notes_bmarks {
  Print::print_report_html_header("Bookmark Categories for Notes", $SAVE);
  print "<p class=\"header01\" align=\"center\">Case Notes</p>\n";
  print_default();
```

```
  print "<center>\n<b>Choose Bookmark</b><br><br>\n| "
       . ($SAVE ? "<a href=\"$NOTES_HTML#default\" " :
                  "<a href=\"$REPORT_URL&view=$Reports::NOTES_ALL#default\" ")
       . "target=\"notes\">&lt;default&gt;</a> | ";

  my @bookmarks = Notes::read_bookmarks();
  if (scalar @bookmarks > 0) {
    foreach my $i (@bookmarks) {
      #make menu of bookmarks
      print ""
         . ($SAVE ? "<a href=\"$NOTES_HTML#$i\" " :
                    "<a href=\"$REPORT_URL&view=$Reports::NOTES_ALL#$i\" ")
         . "target=\"notes\">$i</a> | \n";
    }
  }
  print "</center>\n";

  Print::print_html_footer();
  return;
}

# Show all the case notes
sub notes_all {
  Print::print_report_html_header("Notes for Case $Args::args{'case'}", $SAVE);

  my %bookmarks;

  my $save_inv = $Args::args{'inv'};
  my @invs = Caseman::read_invest();
  if ( inv_unknown() ) {
    push @invs, "unknown";
  }
  my @hosts = get_hosts();

  print "<center>\n";
  if (scalar @hosts == 0) {
    print "No hosts have been added to the case yet.";
  }

  else {
    foreach my $h (sort @hosts) {
      foreach my $i (@invs) {
       $Args::args{'inv'} = $Args::enc_args{'inv'} = $i;
       $::host_dir = "$::case_dir$h/";
       my $nfile = Notes::investig_notes_fname();

       if (-e $nfile && !(-z $nfile) ) {
         open (NOTES, "<$nfile") or die "Could not open $nfile: $!";
         my @inv_note;
         my $str = "";
         my $bmarkname = "default";

         while (<NOTES>) {
           if (/^Bookmark:\s+(.*)$/) {
             $bmarkname = $1 if ($1 ne "");
           }
           elsif (/^\-+$/) {
             $str =~ s/\n/<br>/g;  #replace newlines w/ <br> tag

             push @inv_note, $h;     #hostname
             push @inv_note, $i;     #investigator name
             push @inv_note, $str;   #note
```

```
            push @{ $bookmarks{$bmarkname} }, [@inv_note];

            #reset stuff
            @inv_note = ();
            $str = "";
            $bmarkname = "default";
          }
          else {
            $str .= $_;
          }
        }
       }
      }
    }

    #print all the notes organized by bookmark name
    if (scalar keys %bookmarks != 0) {
      foreach my $b (sort keys %bookmarks) {
       print "<table width=\"350\" border=\"1\" cellspacing=\"0\">\n"
           . "<tr align=\"center\" bgcolor=\"$BGCOLOR\">\n"
           . "<td colspan=\"2\"><a name=\"$b\"><b>$b</b></a></td></tr>\n";

       foreach my $n (@{ $bookmarks{$b} }) {
         print "<tr align=\"center\" bgcolor=\"$::BACK_COLOR_TABLE\">\n"

         #Hostname
             . "<td width=\"50%\">$n->[0]</td>\n<td>"

         #Investigator Name
             . ($SAVE ? "<a href=\"../$INVDIR/$n->[1]/$NOTES_HTML#$n->[0]\" " :
                "<a href=\"$REPORT_URL&view=$Reports::INV_NOTES&luinv=$n-
>[1]#$n->[0]\" ")
             . "target=\"content\">$n->[1]</a></td>\n"

         #Contents of Note
             . "<tr bgcolor=\"$::BACK_COLOR\">\n"
             . "<td colspan=\"2\" align=\"left\">\n$n->[2]</td></tr>\n";
       }
       print "</table><p> </p>\n";
      }
    }
    else {
      print "No Notes have been created for this case<br>\n";
    }
  }
  print "</center>\n";
  Print::print_html_footer_frameset();

  #restore investigator that is viewing case report
  $Args::args{'inv'} = $Args::enc_args{'inv'} = $save_inv;
  return 0;
}

# Show the notes for a specific investigator
sub inv_notes {
  my $inv = shift @_;
  Print::print_report_html_header("Notes for Investigator $inv", $SAVE);
  print "<table width=\"100%\" border=\"0\">\n<tr><td>\n"
      . "<p class=\"header01\" align=\"center\">Investigator Notes for
$inv</p>\n";
  print_default();
```

```
  my @hosts = get_hosts();

  my $count = 0;
  my $bgcolor = $::BACK_COLOR;

  foreach my $h (sort @hosts) {
    print "<table width=\"100%\" border=\"0\"><tr><td colspan=\"2\">\n"
       . "<p class=\"header02\">"
       . "<a name=\"$h\">Host \"$h\"</a></p></td></tr>\n";

    my $notesfile = "$::case_dir/$h/$::LOGDIR/$inv.notes";

    if ( (-e $notesfile) && (! -z $notesfile) ) {
      open (NOTES, "$notesfile") or die "Could not open $notesfile: $!";

      print "<tr align=\"left\" valign=\"top\"><td>\n"
         . "<table border=\"1\" width=\"300\"><tr align=\"left\"
valign=\"top\"><td bgcolor=\"$bgcolor\">";

      while(<NOTES>) {
       if (/^\-+$/) {
         if ($count++ % 2 == 0) {
           $bgcolor = $::BACK_COLOR_TABLE;
         }
         else {
           $bgcolor = $::BACK_COLOR;
         }
         print "</td></tr><tr align=\"left\" valign=\"top\"><td
bgcolor=\"$bgcolor\">\n";
       }
       else {
         print "$_<br>";
       }
      }
      print "</td></tr></table></td></tr>\n";
    }
    print "<tr><td><p> </p></td></tr></table>\n";
  }

  print "</td></tr></table>\n";
  Print::print_html_footer_frameset();
  return 0;
}

# Generate html frames page for displaying case sequencer events
sub seq_frame {
  # begin printing html frames
  Print::print_html_header_frameset("Sequencer Events for Case
$Args::args{'case'}", $SAVE);
  print "<frameset rows=\"250,100%\" framespacing=\"0\" border=\"0\">\n";

  if ($SAVE) {
    print "<frame src=\"$BMARKS_HTML\">\n"
       . "<frame src=\"$EVENTS_HTML\" name=\"timeline\">\n</frameset>\n";
  }
  else {
    print "<frame src=\"$REPORT_URL&view=$Reports::SEQ_BMARKS\">\n"
       . "<frame src=\"$REPORT_URL&view=$Reports::SEQ_EVENTS\" "
       . "name=\"timeline\">\n</frameset>\n";
  }
  print <<EOF;
```

```
<NOFRAMES>
  <center>
    Autopsy requires a browser that supports frames.
  </center>
</NOFRAMES>
EOF

  Print::print_html_footer_frameset();
  return;
}

# Display in frame list of sequencer event bookmarks
sub seq_bmarks {
  Print::print_report_html_header("Bookmark Categories for Events", $SAVE);
  print "<p class=\"header01\" align=\"center\">Sequencer Events</p>\n";
  print_default();

  print "<center>\n<b>Choose Bookmark</b><br><br>\n"
      . ($SAVE ? "<a href=\"$EVENTS_HTML\" target=\"timeline\">" :
          "<a href=\"$REPORT_URL&view=$Reports::SEQ_EVENTS\"
target=\"timeline\">")
      . "All Events</a><br>\n| "
      . ($SAVE ? "<a href=\"default-events.html\" " :
             "<a href=\"$REPORT_URL&view=$Reports::SEQ_EVENTS&bmark=\"\"\" ")
      . "target=\"timeline\">default</a> | \n";

  my @bookmarks = Notes::read_bookmarks();
    if (scalar @bookmarks > 0) {
      foreach my $b (@bookmarks) {
       #make menu of bookmarks
       print ""
           . ($SAVE ? "<a href=\"$b-events.html\" " :
                   "<a href=\"$REPORT_URL&view=$Reports::SEQ_EVENTS&bmark=$b\"
")
           . "target=\"timeline\">$b</a> | \n";
      }
    }

  print "</center>\n";

  Print::print_html_footer();
  return;
}

# Show the sequencer events
sub seq_events {
  #filter out results by bookmark if given
  my $bmark_filter = shift;

  Print::print_report_html_header("Sequencer Event Timeline", $SAVE);
  print "<table width=\"100%\" border=\"0\"><tr><td>\n" .
       "<center>\n";

  my %events = get_seq_events($bmark_filter);
  if (scalar(@{ $events{'sorted'} }) > 0) {
    # Table and header
    print "<table width=\"95%\" border=1 cellspacing=0>\n<tr
bgcolor=\"$BGCOLOR\" align=\"center\">\n"
      . "<td>Bookmark</td>\n<td>Date & Time</td>\n<td>Source</td>\n"
      . "<td>Event & Note</td>\n<td>Host & Investigator</td></tr>\n";
  }
  else {
```

```
    print "There are no sequencer events.\n";
  }

  # Cycle through the sorted events
  my $row = 0;
  foreach my $i (@{ $events{'sorted'} }) {

    # Alternate row colors
    if (($row % 2) == 0) {
      print "<tr bgcolor=\"$::BACK_COLOR\" align=\"center\">\n";
    }
    else {
      print "<tr bgcolor=\"$::BACK_COLOR_TABLE\" align=\"center\">\n";
    }
    # Bookmark
    if ( (defined $events{bmark}[$i]) && ( $events{bmark}[$i] ne "") ) {
      print "<td valign=\"top\">\n"
       .   "<a name=\"$events{bmark}[$i]\">$events{bmark}[$i]</a></td>\n";
    }
    else {
      print "<td> </td>\n";
    }

    # Date & Time
    print "<td valign=top>\n"
       .
"<b>$::d2m[$events{mon}[$i]] $events{day}[$i], $events{year}[$i]"
       . " $events{hour}[$i]:$events{min}[$i]:$events{sec}[$i]</b></td>"
       . "<td valign=top>\n";

    # If there is as filename, then we will show it
    if ($events{fname}[$i] ne "") {
      print "$events{fname}[$i]\n";
    }
    # Display the meta value if there was no name
    elsif ( ($events{vol}[$i] ne "") && (defined $events{meta}[$i]) &&
($events{meta}[$i] ne "") )
      {
       $Args::args{'host'} = $events{host}[$i];
       $::host_dir = "$::case_dir$events{host}[$i]/";
       Caseman::read_host_config();
       my $ftype = $Caseman::vol2ftype{$events{vol}[$i]};
       print "$Fs::meta_str{$ftype}: $events{meta}[$i]\n";
      }
      # Otherwise, just give the source type
    else {
      print "$events{type}[$i]\n";
    }
    #file type
    print "<br>$events{filetype}[$i]";

    print "</td>\n";

    # Print the actual note
    $events{note}[$i] = " " if ($events{note}[$i] eq "");
    print "<td align=left valign=top>$events{note}[$i]</td>\n";

    # Print investigator and host
    print "<td
valign=\"top\">$events{host}[$i]<br>$events{invest}[$i]</td></tr>\n";

    $row++;
```

```
  }
  print "</table>\n";
  Print::print_html_footer();
  return;
}

# Generate html frames page for displaying keyword searches
sub kword_frame {
  # begin printing html frames
  Print::print_html_header_frameset("Keyword Searches for Case
$Args::args{'case'}", $SAVE);
  print "<frameset rows=\"350,100%\" framespacing=\"0\" border=\"0\">\n";

  if ($SAVE) {
    print "<frame src=\"$LIST_HTML\">\n"
       . "<frame src=\"choose.html\" name=\"results\">\n</frameset>\n";
  }
  else {
    print "<frame src=\"$REPORT_URL&view=$Reports::KWORD_LIST\">\n"
       . "<frame src=\"$REPORT_URL&view=$Reports::KWORD_RESULTS\" "
       . "name=\"results\">\n</frameset>\n";
  }
  print <<EOF;
<NOFRAMES>
  <center>
    Autopsy requires a browser that supports frames.
  </center>
</NOFRAMES>
EOF

  Print::print_html_footer_frameset();
  return;
}

# Show list of keyword searches
sub kword_list() {
  Print::print_report_html_header("List of Keyword Searches", $SAVE);
  print "<p class=\"header01\" align=\"center\">Keyword Searches</p>\n";
  print_default();
  print "<center>\n";

  my %searches = get_searches();

  if (keys(%searches) > 0) {
    print "<table border=\"1\" cellspacing=\"0\" width=\"90%\">\n"
       . "<tr bgcolor=\"$BGCOLOR\" align=\"center\">\n"
       . "<td>Keyword</td>\n<td>Hits</td>\n<td>Type</td>\n"
       . "<td>vol</td>\n<td>Host</td></tr>\n";
  }

  else {
    print "No keyword searches have been made.";
  }
  my $row = 0;

  foreach my $keyword (sort keys %searches) {
    foreach my $i (@{ $searches{$keyword} }) {
      my $hits = $i->[0];
      my $type = $i->[1];
      my $grep = $i->[2];
      my $host = $i->[3];
      my $sname = $i->[4];
```

```
      my $vol = $i->[5];
      my $fname = $i->[6];
      my $html_keyword = Args::url_encode($keyword);

      # Alternate row colors
      if (($row++ % 2) == 0) {
       print "<tr bgcolor=\"$::BACK_COLOR\">\n";
      }
      else {
       print "<tr bgcolor=\"$::BACK_COLOR_TABLE\">\n";
      }
      print "<td>"
         . ($SAVE ? "<a href=\"$html_keyword-$sname-$type.html\"
target=\"results\">" :
"<a
href=\"$REPORT_URL&view=$Reports::KWORD_RESULTS&host=$host&vol=$vol&fname=$fnam
e\" target=\"results\">")
         . "$keyword</td>\n<td>$hits</td>\n"
         . "<td>$type</td>\n<td>$vol   $sname</td>\n"
         . "<td>$host</td>\n</tr>\n";
    }
  }
  print "</table>\n";
  Print::print_html_footer();
  return 0;
}

# Show the results for a keyword search
sub kword_results {
  Print::print_report_html_header("Results of Keyword Search", $SAVE);

  my $fname = "";
  $fname = $Args::args{'fname'} if exists $Args::args{'fname'};

  if ($fname ne "") {
    unless (open(SRCH, "$fname")) {
      print "Error opening search file: $fname\n";
      return 1;
    }
    my $host = Args::get_host('host');
    $::host_dir = "$::case_dir$host/";
    Caseman::read_host_config();
    my $vol = Args::get_vol('vol');

    my $ftype     = $Caseman::vol2ftype{$vol};
    my $addr_str  = $Fs::addr_unit{$ftype};
    my $grep_str  = "";
    my $grep_flag = "";
    my $cnt       = 0;
    my $type      = 0;    # ASCII iis 0 and Unicode is 1

    my $prev = -1;

    while (<SRCH>) {
       # The first line is a header
       if ($. == 1) {
          if (/^(\d+)\|(.*?)?\|(.*)$/) {
             $cnt       = $1;
             $grep_flag = $2;
             $grep_str  = $3;
             $type      = 0;
          }
```

```
          else {
             print "Error pasing header of search file: $fname\n";
             close(SRCH);
             return 1;
           }

           if ($grep_str =~ /^(.*?)\|unicode$/) {
             $grep_str = $1;
             $type     = 1;
           }
           elsif ($grep_str =~ /^(.*?)\|ascii$/) {
             $grep_str = $1;
           }

           my $grep_str_html = Print::html_encode($grep_str);
           print "<hr>\n";
           if ($type == 0) {
             print "<a name=\"ascii\">\n";
           }
           else {
             print "<a name=\"unicode\">\n";
           }

           if ($cnt == 0) {
             print "<b><tt>$grep_str_html</tt> was not found</b><br>\n";
           }
           elsif ($cnt == 1) {
              print
"<b>1 occurrence of <tt>$grep_str_html</tt> was found</b><br>\n";
           }
           else {
             print
"<b>$cnt occurrences of <tt>$grep_str_html</tt> were found</b><br>\n";
           }

           print "Search Options:<br>\n";
           if ($type == 0) {
             print "  ASCII<br>\n";
           }
           else {
             print "  Unicode<br>\n";
           }

           if ($grep_flag =~ /\-i/) {
             print "  Case Insensitive<br>\n";
           }
           else {
             print "  Case Sensitive<br>\n";
           }
           if ($grep_flag =~ /\-E/) {
             print "  Regular Expression<br>\n";
           }

           print "<hr>\n";

           #if ($cnt > 1000) {
           #  print "There were more than <U>1000</U> hits.<br>\n";
           #  print "Please revise the search to a managable amount.\n";
           #  print
           #  "<p>The $cnt hits can be found in: <tt>$fname</tt><br>\n";
           #  close(SRCH);
           #  return 0;
```

```
            #}
            next;
        }

        my $meta   = "";
        my $src    = "";
        my $blk    = "";
        my $off    = "";
        my $str    = "";

        if (/^(\d+)\|(\d+)\|(.*)\|(\d+)\|(.*)?$/) {
            $blk    = $1;
            $off    = $2;
            $str    = $3;
            $meta   = $4;
            $src    = $5;
        }

        elsif (/^(\d+)\|(\d+)\|(.*)?$/) {
            $blk = $1;
            $off = $2;
            $str = $3;
        }

        else {
            print "Error parsing results: $_\n";
            close(SRCH);
            return 1;
        }

        if ($blk != $prev) {
          print "<br>\n$addr_str $blk<br>";

          #print meta info
          print
"$Fs::meta_str{$ftype}: $meta<br>" if ($meta ne "");

          if ($src ne "") {
            my $root = $Caseman::vol2mnt{$vol};

            #Print file info and make it red if it is deleted
            if ($src =~ /^(\*)\s+\/*(.*)$/) {
              Print::print_output("File:
<font color=\"$::DEL_COLOR[0]\">$root$2</font> (deleted)<br>\n");
            }

            # If it starts with a '/' then it must be a file
            elsif ($src =~ /^\/(.*)$/) {
              Print::print_output("File: $root$src<br>\n");
            }
            # must be an error
            else {
              print "File: $src<br>\n";
            }
          }
          $prev = $blk;
        }

        my $occ = $. - 1;
        if ($str ne "") {
           $str = Print::html_encode($str);
           print "$occ: $off (<tt>$str</tt>)<br>\n";
```

```
      }
      else {
         print "$occ: $off<br>\n";
      }
    }

    close(SRCH);
    return 0;
  }

  else {
    print "<center>\n<b>Choose a keyword listed above to view the results"
       . "</b><br>\n</center>\n";
  }

  Print::print_html_footer();
  return 0;
}

# Show a summary of the file categories; how many files for each category
sub file_cat_sum {
  Print::print_report_html_header("File Categories", $SAVE);
  print "<p class=\"header01\" align=\"center\">File Categories</p>\n";
  print_default();

  my @hosts = get_hosts();
  my %hosthash;
  my @categories;
  my $getCat = 1; #flag meaning to parse category names

  #scan for sorter directories in each host
  foreach my $h (sort @hosts) {
    my $output_dir = "$::case_dir$h/$::DATADIR/";
    opendir OUTPUT, $output_dir
      or die "Can't open directory $output_dir";

    my %volhash;
    foreach my $sort_dir (readdir OUTPUT) {
      next unless ($sort_dir =~ /^sorter\-(vol\d+)$/);


      if (-e "$output_dir/$sort_dir/index.html") {
       my $vol = $1;

       # parse the index.html file
       open INDEX, "<$output_dir/$sort_dir/index.html"
         or die "Can't open sorter index file ($sort_dir/index.html)";

       while (<INDEX>) {
         next unless(/^\s*(<.*>)?(<.*>)?([\w\s\'\-]+)(<.*>)?\s\((\d+)\)$/);
         push @categories, $3 if $getCat;
         push @{ $volhash{$vol} }, $5;
       }
       $getCat = 0 if ($getCat == 1); #no need to keep parsing category names
       close(INDEX);
       %{ $hosthash{$h} } = %volhash;
      }
    }
    close(OUTPUT);
  }

  if (keys(%hosthash) > 0) {
```

```
    print "<center><table border=\"0\" cellspacing=\"0\">\n"
       . "<tr bgcolor=\"$BGCOLOR\" align=\"center\">\n"
       . "<td> </td>\n";
    #print row of host names
    foreach my $h (sort keys %hosthash) {
      print "<td colspan=\"" . keys (%{ $hosthash{$h} }) .
"\"><b>$h</b></td>\n";
    }
    print "</tr>\n";

    #print row of mount names
    print "<tr align=\"center\"><td> </td>\n";
    foreach my $h (sort keys %hosthash) {
      foreach my $vol (sort keys %{ $hosthash{$h} }) {
       $::host_dir = "$::case_dir$h/";
       $Args::args{'host'} = $Args::enc_args{'host'} = $h;
       Caseman::read_host_config();
       print "<td>$Caseman::vol2mnt{$vol}</td>\n"
      }
    }
    print "</tr>\n";

    #print table of File Category Names
    my $row = 0;
    my $bgcolor;
    print "<tr><td><table border=\"0\">\n";
    foreach my $cat (@categories) {
      if ($row++ % 2 == 0) {
       $bgcolor = $::BACK_COLOR_TABLE;
      }
      else {
       $bgcolor = $::BACK_COLOR;
      }
      print "<tr align=\"right\" bgcolor=\"$bgcolor\"><td>";
      if ($cat eq "Files" || $cat eq "Files Skipped" ||
       $cat eq "Extension Mismatches" || $cat eq "Categories")
      {
       print "<b>$cat</b></td></tr>\n";
      }
      else {
       print "$cat</td></tr>\n";
      }
    }
    print "</table></td>\n";

    #print table of file category values for each vol
    foreach my $h (sort keys %hosthash) {
      foreach my $vol (sort keys %{ $hosthash{$h} } ){
       $row = 0;
       print "<td><table border=\"0\" width=\"60\">\n";
       my $cnt = 0;
       foreach my $val (@{ $hosthash{$h}{$vol} }) {
         if ($row++ % 2 == 0) {
           $bgcolor = $::BACK_COLOR_TABLE;
         }
         else {
           $bgcolor = $::BACK_COLOR;
         }
         $categories[$cnt] = "mismatch"
           if ($categories[$cnt] eq "Extension Mismatches");
         my $sorter_file = "$::case_dir$h/$::DATADIR/sorter-$vol/" .
$categories[$cnt] . ".html";
```

```
        print "<tr align=\"center\" bgcolor=\"$bgcolor\"><td>";
        if (-e "$sorter_file") {
          print ""
             . ($SAVE ? "<a href=\"$h-$vol-$categories[$cnt].html\">" :
                  "<a
href=\"$REPORT_URL&view=$Reports::FILE_CAT&host=$h&vol=$vol&cat=$categories[$cn
t]\">")
             . "$val</a></td></tr>\n";
        }
        else {
          print "$val</td></tr>\n";
        }
        $cnt++;
       }
       print "</table></td>\n";
      }
    }
    print "</table></center>\n";
  }

  else {
    print "<center>File types have not been sorted.<br>This feature is only
available for "
       . "non-raw disk images.\n</center>";
  }
  Print::print_html_footer();
  return 0;
}

# Show the details for the files in a specified category
sub file_cat {
  my $cat = $Args::args{'cat'};
  Print::print_report_html_header("File Categories: $cat", $SAVE);
  print "<p class=\"header01\" align=\"center\">File Categories: $cat</p>\n";
  print_default();

  my $host = Args::get_host('host');
  $::host_dir = "$::case_dir$host/";
  $Args::args{'host'} = $Args::enc_args{'host'} = $host;
  Caseman::read_host_config();

  my $vol = Args::get_vol('vol');


  print "<p class=\"header02\">$host   $Caseman::vol2mnt{$vol}</p>\n";
  my $sorter_file = "$::host_dir/$::DATADIR/sorter-$vol/$cat.html";

  if (-e "$sorter_file") {
    open FILE, "$sorter_file"
      or die "Can't open file $sorter_file: $!";

    while (<FILE>) {
      next if (/^<\/?HTML>/i || /^<\/?HEAD>/i || /^<\/?TITLE>/i ||
/^<\/?BODY>/i);
      print $_;
    }
  }
  Print::print_html_footer();
  return 0;
}
```

```
# Generate a timeline of graphic images that have been bookmarked; display
thumbnails
sub thumbs {
  my $case = Args::get_case();
  Print::print_report_html_header("Picture Thumbnails", $SAVE);
  print "<p class=\"header01\" align=\"center\">Picture Thumbnails</p>\n";
  print_default();
  print "<center>To view thumbnails be sure to create a sequencer event "
      . "for the 'interesting' images.<p><br></p>\n<table border=\"0\"
cellpadding=\"8\">\n";

  my %events = get_seq_events();

  my $count = 0;
  my $row = 0;
  print "<tr>";

  foreach my $i (@{ $events{'sorted'} }) {
    if (($events{filetype}[$i] =~ /image data/) || ($events{filetype}[$i] =~
/PC bitmap data/)) {

      $Args::args{'host'} = $Args::enc_args{'host'} = $events{host}[$i];
      $::host_dir = "$::case_dir$events{host}[$i]/";
      Caseman::read_host_config();
      my $mnt = $Caseman::vol2mnt{$events{vol}[$i]};
      my $dir = $1 if ($events{fname}[$i] =~ /^$mnt(.*)$/);
      my $filename = $1 if ($dir =~ /\/?([^\/]*)$/);
      my $date =
"$::d2m[$events{mon}[$i]] $events{day}[$i], $events{year}[$i]"
             . " $events{hour}[$i]:$events{min}[$i]:$events{sec}[$i]";
      my $note = "";
      my $time = "";
      if ($events{note}[$i] =~ /^(\[.?\-Time\])(.*)$/) {
       $time = $1 if (defined $1);
       $note = $2 if (defined $2);
      }

      if ($count++ % 4 == 0) {
       print "</tr><tr align=\"center\"><td>" . ++$row . "</td>\n"
      }
      print "<td width=\"150\">"
         . ($SAVE ? "<a href=\"$THUMBSDIR/$events{meta}[$i]-$filename.html\">" :
             "<a
href=\"$::PROGNAME?mod=$::MOD_REPORTS&view=$Reports::PIC_DETAILS&"
           . "case=$case&host=$events{host}[$i]&inv=$Args::args{'inv'}&"
           . "vol=$events{vol}[$i]&meta=$events{meta}[$i]&"
           . "dir=$dir&filetype=$events{filetype}[$i]&time=$time&date=$date"
           .
"&note=$note&luinv=$events{invest}[$i]&bookmark=$events{bmark}[$i]"
           . "&cell_mode=2&recmode=1\">")

        . ($SAVE ? "<img src=\"$THUMBSDIR/$events{meta}[$i]-$filename\"" :
            "<img
src=\"$::PROGNAME?mod=$::MOD_APPVIEW&view=$Appview::CELL_CONT&"
          .
"case=$case&host=$events{host}[$i]&vol=$events{vol}[$i]&meta=$events{meta}[$i]&
"
          . "dir=" . Args::url_encode($dir)
          . "&cell_mode=2&recmode=1\"")

        . " width=\"150\" height=\"150\" alt=\"$filename\">"
        . "<br>$filename</a><br>$time $date<br></td>\n";
```

```
    }
  }

  print "</table>\n";
  Print::print_html_footer();
  return 0;
}

# Show full-size image, and bookmark information
sub pic_details {
  my $case = Args::get_case();
  my $host = Args::get_host();
  $Args::args{'host'} = $Args::enc_args{'host'} = $host;
  $::host_dir = "$::case_dir$host/";
  Caseman::read_host_config();

  my $vol  = Args::get_vol('vol');
  my $meta = Args::get_meta('meta');
  my $mnt  = $Caseman::vol2mnt{$vol};
  my $dir  = Args::get_dir();
  my $filename = $1 if ($dir =~ /\/?([^\/]*)$/);
  my $filetype = $Args::args{'filetype'};
  my $date = $Args::args{'date'};
  my $inv = $Args::args{'luinv'};
  my $note = $Args::args{'note'};
  my $bmark = Args::get_bmark();
  my $time = $Args::args{'time'};

  Print::print_report_html_header("Picture Details", $SAVE);
  print "<p class=\"header01\" align=\"center\">Picture Details</p>\n";
  print_default();

  print "<center>\n"
      . ($SAVE ? "<img src=\"$meta-$filename\"" :
         "<img src=\"$::PROGNAME?mod=$::MOD_APPVIEW&view=$Appview::CELL_CONT&"
       . "case=$case&host=$host&vol=$vol&meta=$meta&dir=$dir"
       . "&cell_mode=2&recmode=1\"")

      . " alt=\"$filename\"><br>$mnt$dir\n<br>$filetype\n<p>$time $date\n<p>\n"
      . "Bookmarked by $inv\n<br>Bookmark Name: $bmark<p>\n"
      . "Note:<br>$note\n</center>";

  Print::print_html_footer();
  return 0;
}

# Show and link to any of the individual reports in the $REPDIR directory
sub indv_rep {
  Print::print_report_html_header("Individual Reports", $SAVE);
  print "<p class=\"header01\" align=\"center\">Individual Reports</p>\n";
  print_default();

  print "<center>Individual reports must be saved to "
      . "$::case_dir" . "<i>host-dir</i>/$::REPDIR</center>\n";

  my %reports = get_reports();
  if ( keys %reports < 1 ) {
    print "<p align=\"center\"><br>There are no individual reports</p>\n";
  }

  foreach my $host (sort keys %reports) {
    print "<p class=\"header02\">Host $host</p>\n<ul>\n";
```

```
    foreach my $rep (sort @{ $reports{$host} }) {
      print "<li>"
         . ($SAVE ? "<a href=\"$host/$rep.html\">" :
              "<a href=\"$REPORT_URL&host=$host&view=$Reports::PRINT_2HTML&"
                 . "show=report&report=" . Args::url_encode($rep) . "\">")
         . "$rep</a></li>\n";
    }
    print "</ul>\n";
  }

  Print::print_html_footer();
  return 0;
}

# END ANALYSIS INFORMATION SECTION
################################################################################

# Create a directory for the case report
sub mk_htmldir {
  my $htmldir = shift;
  unless (-d "$htmldir") {
    mkdir("$htmldir", 0775) or die "Can't create directory $htmldir: $!";
  }
}

# Create html file for case report
sub mk_htmlfile {
  my $htmlfile = shift;
  open(STDOUT, "+>$htmlfile")
    or die "Can't open $htmlfile";
}

# Convert various text-based logs and files to html format
sub print_2html {
  my $title = "";
  my $header = "";
  my $str = "";
  my $name = shift @_;
  my $file;

  $::host_dir = "$::case_dir/$Args::args{'host'}/" if exists
$Args::args{'host'};

  if ($name eq "caselog") {
    $title = "History for Case $Args::args{'case'}";
    $header = "Case History";
    $file = $::case_dir . "case.log";
  }

  elsif ($name eq "hostlog") {
    my $h = $Args::args{'host'};
    $title = "History for Host $h";
    $header = "History for Host $h";
    $file = "$::case_dir/$h/$::LOGDIR/host.log";
  }

  elsif ($name eq "invlog") {
    my $inv = $Args::args{'luinv'};
    my $host = $Args::args{'host'};
    $title = "$inv\'s History on $host";
    $header = "$inv\'s History on $host";
    $file = "$::case_dir$host/$::LOGDIR/$inv.log";
```

```
  }
  elsif ($name eq "invcomm") {
    my $inv = $Args::args{'luinv'};
    my $host = $Args::args{'host'};
    $title = "$inv\'s Command History on $host";
    $header = "$inv\'s Command History on $host";
    $file = "$::case_dir$host/$::LOGDIR/$inv.exec.log";
  }
  elsif ($name eq "report") {
    my $rep = $Args::args{'report'};
    my $host = Args::get_host();
    open REPORT, "$::case_dir$host/$::REPDIR/$rep"
      or die "Can't open report $::case_dir$host/$::REPDIR/$rep: $!";
    $title = <REPORT>;
    $header = $title;
    $file = "$::case_dir$host/$::REPDIR/$rep";
  }
  open FILE, "$file" or die "Could not open $file: $!";

  while(<FILE>) {
    $str .= "<p class=\"header04\">" . $_ . "<br></p>\n";
  }

  Print::print_report_html_header("$title", $SAVE);
  print "<table width=\"100%\" border=\"0\"><tr><td>\n" .
       "<p class=\"header01\" align=\"center\">$header</p>\n";
  print_default();
  print $str;
  Print::print_html_footer();
  return 0;
}

# Print the default message on each page of the report
sub print_default {
  print "<p class=\"small\">Case Report generated using Autopsy v$aut_ver, The
Sleuth Kit v$tsk_ver<br>\n";

  #print report created time
  my $date = localtime();
  print "$date\n<br>\n";

  #print investigator
  print "Investigator: $Args::args{'inv'}<br>\n";

  #print case
  print "Case: $Args::args{'case'}</p><hr>\n";

  return 0;
}

# Get the list of all the hosts in a case
sub get_hosts {
  my @hosts;
  opendir HOSTS, $::case_dir or die "Can't open $::case_dir directory: $!";
  foreach my $h (readdir HOSTS) {
    next if (($h eq '.') || ($h eq '..'));
    my $hfile = Caseman::host_config_fname($h);
    push @hosts, $h
      if ((-d "$::case_dir" . "$h") && (-e "$hfile"));
  }
  closedir HOSTS;
```

```
  return @hosts;
}

# Read the md5 value and volume name pairs from the md5 file
sub get_md5s {
  $::host_dir = "$::case_dir$Args::args{'host'}/";
  my %md5hash;

  #Read MD5 Values
  if (-e "$::host_dir/md5.txt") {
    open(MD5, "$::host_dir/md5.txt")
       or die "Cannot read md5 file\n";
    while (<MD5>) {
      if (/^($::REG_MD5)\s+(.*)$/o) {
       $md5hash{"$2"} = $1;
      }
    }
  }
  return %md5hash;
}

# Read and sort all the sequencer events for a case and store to a hash of
arrays
sub get_seq_events {
  my $bmark_filter = shift;
  my @inv = Caseman::read_invest();
  if ( inv_unknown() ) {
    push @inv, "unknown";
  }
  my @hosts = get_hosts();

  my $cnt = 0;

  # Read the sequencer file into arrays that will be sorted
  my %events;
  my (@entry, @sorted);
  my (
      @bmark, @year, @mon,   @day,  @hour, @min,  @sec, @host,
      @vol,  @fname, @meta, @data, @type, @note, @invest, @filetype
     );
  foreach my $i (@inv) {
    foreach my $h (@hosts) {
      my $seqfile = "$::case_dir$h/$::LOGDIR/$i.seq.notes";
      if (-e $seqfile) {
       open NOTES, "$seqfile" or die "Can't open log: $seqfile";
       while (<NOTES>) {
           unless (
/^'(\d+)','(\d+)','(\d+)','(\d+)','(\d+)','(\d+)','($::REG_HOST)','($::REG_VNAM
E)?','(.*?)','($::REG_META)?','(\d+)?','([\w\s]+)','(.*?)',?'?(.*?)'?,?'?($::RE
G_BMARK)?'?\n/
                  )
             {
              Print::print_err("Error parsing sequence event entry: $_");
             }
         $bmark[$cnt]   = "";
         $bmark[$cnt]   = $15 if (defined $15);

         if ( ! (defined $bmark_filter) || $bmark_filter eq $bmark[$cnt] ) {
           $year[$cnt]  = $1;
           $mon[$cnt]   = $2;
           $day[$cnt]   = $3;
           $hour[$cnt]  = $4;
```

```
            $min[$cnt]   = $5;
            $sec[$cnt]   = $6;
            $host[$cnt]  = $7;
            $vol[$cnt]   = "";
            $vol[$cnt]   = $8 if (defined $8);
            $fname[$cnt] = $9;
            $meta[$cnt]  = "";
            $meta[$cnt]  = $10 if (defined $10);
            $data[$cnt]  = $11;
            $type[$cnt]  = $12;
            $note[$cnt]  = $13;
            $filetype[$cnt] = $14;

            #get filetype if located in seq.notes file
            if (defined $14 && $14 ne "") {
              $filetype[$cnt] = $14;
            }
            elsif (defined $vol[$cnt] && $vol[$cnt] ne ""
                 && defined $meta[$cnt] && $meta[$cnt] ne "")
            {
              #determine filetype if reading old autopsy seq.notes files
              $h = $1 if ($h =~ /($::REG_HOST)/);
              $Args::args{'host'} = $Args::enc_args{'host'} = $h;
              $::host_dir = "$::case_dir$h/";
              Caseman::read_host_config();

              my $ftype = $Caseman::vol2ftype{$vol[$cnt]};
              my $offset = $Caseman::vol2start{$vol[$cnt]};
              my $imgtype = $Caseman::vol2itype{$vol[$cnt]};
              my $img = $Caseman::vol2path{$vol[$cnt]};

              local *OUT;
              die "Can't open pipe"
               unless defined(my $pid = open(*OUT, "-|"));
                 if (! $pid) {
                   $| = 1;
    exec("'$::TSKDIR/icat' -f $ftype -r  -o $offset -i $imgtype $img
$meta[$cnt] | '$::FILE_EXE' -z -b -")
      or die "Cannot exec command: $!";
                 }
              $filetype[$cnt] = Exec::read_pipe_line(*OUT);
              close(OUT);

              $filetype[$cnt] = ""
               if ( (!defined $filetype[$cnt]) );

              chomp $filetype[$cnt]; #get rid of newline
            }

            $invest[$cnt] = $i;
            $entry[$cnt] = $cnt;

            $cnt++;
          }
        }

        close(NOTES);
       }
     }
   }

   if (scalar @entry > 0) {
```

```
    # Sort the values by date, source, and then note
    @sorted = sort {
      $year[$a] <=> $year[$b]
       or $mon[$a] <=> $mon[$b]
         or $day[$a] <=> $day[$b]
           or $hour[$a] <=> $hour[$b]
             or $min[$a] <=> $min[$b]
              or $sec[$a] <=> $sec[$b]
                or lc($invest[$a]) cmp lc($invest[$b])
                  or lc($type[$a]) cmp lc($type[$b])
                    or lc($note[$a]) cmp lc($note[$b])
                  } @entry;
  }
  %events = (
               sorted   => [@sorted],
               bmark    => [@bmark],
               year     => [@year],
               mon      => [@mon],
               day      => [@day],
               hour     => [@hour],
               min      => [@min],
               sec      => [@sec],
               host     => [@host],
               vol      => [@vol],
               fname    => [@fname],
               meta     => [@meta],
               data     => [@data],
               type     => [@type],
               note     => [@note],
               invest   => [@invest],
               filetype => [@filetype],
           );
  return %events;
}

# Read and sort all the keyword searches for a case and store to a hash of
arrays
sub get_searches {
  my @hosts = get_hosts();
  my %searches;

  foreach my $h (sort @hosts) {
    $::host_dir = "$::case_dir$h/";
    $Args::args{'host'} = $Args::enc_args{'host'} = $h;
    my %sname2vol = get_snames();

    #scan output dir for .srch files
    opendir OUTPUT, "$::host_dir/$::DATADIR/"
       or die "Can't open $::host_dir/$DATADIR directory: $!";

    foreach my $srch_name (readdir OUTPUT) {
       if ($srch_name =~ /^(.*)-\d+\.srch$/) {
           my $sname = $1;
           my $vol = $sname2vol{$sname};
           my $fname = "$::host_dir/$::DATADIR/$srch_name";
           unless (open(SRCH, "$fname")) {
              print "Error opening search file: $fname\n";
              return 1;
           }
           while (<SRCH>) {
              unless (/^(\d+)\|(.*?)?\|(.*)\|(.*)$/) {
                print
```

```
                    "Error parsing header of search file: $srch_name\n";
                  return 1;
                }
                my $cnt = $1;
                my $grep = "";
                $grep = $2 if (defined $2);
                my $str = $3;
                my $type = $4;

                if (length($str) > 32) {
                  $str = substr($str, 0, 32);
                  $str .= "...";
                }
                push @{ $searches{$str} }, [$cnt, $type, $grep, $h, $sname,
$vol,$fname];
                  last;   #only read first line header
                }
                close(SRCH);
             }
       }
    close(OUTPUT);
    }
  return %searches;
}

# Create hash of vol name and vol id pairs
sub get_snames {
    my %sname2vol = ();
    my $host = Args::get_host();

    $::host_dir = "$::case_dir$host/";
    Caseman::read_host_config();

    if (scalar(keys %Caseman::vol2ftype) != 0) {
       foreach my $vol (keys %Caseman::vol2cat) {
          $sname2vol{$Caseman::vol2sname{$vol}} = $vol;
       }
    }
    return %sname2vol;
}

# Read the list of files located in the $REPDIR directory
sub get_reports {
  my %reports;
  my @hosts = get_hosts();
  foreach my $h (@hosts) {
    opendir REPORTS, "$::case_dir$h/$::REPDIR"
      or die "Can't open reports dir: $!";
    foreach my $rep (readdir REPORTS) {
      next if ( ($rep eq ".") || ($rep eq "..") );
      push @{ $reports{$h} }, $rep;
    }
  }
  return %reports;
}

# Check if investigator "unknown" exists
sub inv_unknown {
  my @hosts = get_hosts();
  foreach my $h (@hosts) {
    if (-e "$::case_dir$h/$::LOGDIR/unknown.log") {
       return 1;
```

```
    }
  }
  if ( scalar(my @inv = Caseman::read_invest()) < 1) {
    return 1;
  }
  return 0;
}


1;
```

B. APPSORT.PM

```
2c2
< # $Date: 2005/09/18 22:49:44 $
---
> # $Date: 2005/09/18 22:49:44 $
6a7
> # Regis Friend Cassidy [rfriendcassidy@gmail.com]
38,43c39,48
< $Appsort::FRAME = 1;
< $Appsort::MENU  = 2;
< $Appsort::ENTER = 3;
< $Appsort::RUN   = 4;
< $Appsort::VIEW  = 5;
< $Appsort::BLANK = 6;
---
> $Appsort::FRAME       = 1;
> $Appsort::MENU        = 2;
> $Appsort::ENTER       = 3;
> $Appsort::RUN         = 4;
> $Appsort::VIEW        = 5;
> $Appsort::CAT_FR      = 6;
> $Appsort::VIEW_CAT    = 7;
> $Appsort::VIEW_THUMBS = 8;
> $Appsort::BLANK       = 9;
> $Appsort::CAT_BLANK   = 10;
64a70,72
>     elsif ($view == $Appsort::CAT_BLANK) {
>         return cat_blank();
>     }
84a93,101
>     elsif ($view == $Appsort::CAT_FR) {
>         return cat_fr();
>     }
>     elsif ($view == $Appsort::VIEW_CAT) {
>         return view_cat();
>     }
>     elsif ($view == $Appsort::VIEW_THUMBS) {
>         return view_thumbs();
>     }
112c129
<       . "$Args::baseargs\">\n";
---
>       . "$Args::baseargs\" name=\"menu\">\n";
133a151,168
>
>     # Check if sorter has been run to extract graphic images
>     my $graphic_dir = get_sorter_graphics_dir();
>     if (-e "$graphic_dir/images.html") {
>
>       # Show the links to the thumbnails pages
>       print "<p>View Thumbnails<BR>\n";
>       opendir IMAGES, "$graphic_dir/images";
>       my $count = 1;
>       foreach my $thumbs (sort readdir IMAGES) {
>         if ($thumbs =~ /^thumbs-\d+\.html$/) {
>           print "  <a href=\"$::PROGNAME?mod=$::MOD_APPSORT&"
>               . "view=$Appsort::VIEW_THUMBS&$Args::baseargs&page=$count\" "
>               . "target=\"content\">page $count</a><BR>";
>           $count++;
```

```
>         }
>       }
>     }
346a382,408
>     if ( (exists $Args::args{'sorter_img'})
>             && ($Args::args{'sorter_img'} == 1))
>     {
>         # User needs to refresh page to view links to thumbnails
>         print "<a href=\"$::PROGNAME?mod=$::MOD_APPSORT&view=$Appsort::MENU&"
>             . "$Args::baseargs\" target=\"menu\"><p><center><b>"
>             . "Click here</a> to refresh the menu on the left and view "
>             . "the thumbnails</b></center>";
>     }
>     else {
>       print "<p><center><b>Click the 'View Sorted Files' link to view"
>           . "the results</b></center>\n";
>     }
>
>     Print::print_html_footer();
>     return;
> }
>
> # View Page
> # This routine parses the index.html file produced by sorter
> # to fit the Autopsy format. Links to the results pages for
> # each category are maintained.
> sub view {
>     Print::print_html_header("");
>
>     my $sort_dir = get_sorter_dir();
>
348c410
<         print "<p>Output can be found by viewing:<br>"
---
>       print "<p>Output also saved to:<br>"
351,354c413,416
<         # Print the index.html file from the output
<         print "<hr><center><h3>Results Summary</h3></center>\n";
<         open INDEX, "<$sort_dir/index.html"
<           or die "Can't open sorter index file ($sort_dir/index.html)";
---
>       # Print the index.html file from the output
>       print "<hr><center><h3>Results Summary</h3></center>\n";
>       open INDEX, "<$sort_dir/index.html"
>         or die "Can't open sorter index file ($sort_dir/index.html)";
356,357c418,419
<         while (<INDEX>) {
<             next if ((/^<HTML><HEAD><TITLE>/i)
---
>       while (<INDEX>) {
>         next if ((/^<HTML><HEAD><TITLE>/i)
360,371c422,425
<             # Extract out the symlinks to the categories
<             if (/^\s*<li><a href="\.\/[\w\.]+">([\w\s]+)<\/a>
\((\d+)\)\s*$/i) {
<                 print "<LI>$1 ($2)\n";
<             }
<
<             # Skip the link on the thumbnails link
<             elsif (/^\s*\(<a href=[\"\.\/\w]+>thumbnails<\/A>\)\s*$/) {
<                 print "(thumbnails)\n";
<             }
```

```
<             else {
<                 print "$_";
<             }
---
>         # Extract out the symlinks to the categories
>         if (/^\s*<li><a href="\.\/([\w\.]+)">([\w\s]+)<\/a> \((\d+)\)\s*$/i)
{
>           print "<LI><a
href=\"$::PROGNAME?mod=$::MOD_APPSORT&view=$Appsort::CAT_FR&"
>               . "$Args::baseargs&cat=$1\">$2</a> ($3)\n";
373,374d426
<         close(INDEX);
<     }
375a428,442
>         # Skip the link on the thumbnails link
>         elsif (/^\s*\(<a href=[\"\.\/\w]+>thumbnails<\/A>\)\s*$/) {
>            print "(thumbnails)\n";
>         }
>         else {
>            print "$_";
>         }
>       }
>       close(INDEX);
>     }
>     else {
>       print "<center>Sorter has not been run yet. "
>           . "Click the link 'Sort Files by Type' from the menu</center>";
>     }
>
377c444
<     return;
---
>     return 0;
380,386c447,449
< # View Page
< sub view {
<     Print::print_html_header("");
<     print "<center><h3>File Type Sorting</h3>\n"
<       . "Autopsy does not currently support viewing the sorted files.<br>\n"
<       . "After sorting, you can view the results by opening the following
file:<p>\n";
<     print "<tt>" . get_sorter_dir() . "index.html</tt>";
---
> # category results frameset
> sub cat_fr {
>     Print::print_html_header_frameset("Sorter on $Args::args{'vol'}");
388c451,462
<     Print::print_html_footer();
---
>     print "<frameset rows=\"60%,40%\">\n";
>
>     # File List
>     print "<frame
src=\"$::PROGNAME?mod=$::MOD_APPSORT&view=$Appsort::VIEW_CAT&"
>       . "$Args::baseargs&cat=$Args::args{'cat'}\">\n";
>
>     # File Contents
>     print "<frame
src=\"$::PROGNAME?mod=$::MOD_APPSORT&view=$Appsort::CAT_BLANK&"
>       . "$Args::baseargs\" name=\"file_content\">\n"
>       . "</frameset>\n";
>
```

```
>     Print::print_html_footer_frameset();
391a466,604
> # View Page of Category results
> sub view_cat {
>   Print::print_html_header("sorted $Args::args{'cat'} files");
>
>   my $sort_file = get_sorter_dir() . $Args::args{'cat'};
>
>   if (-e "$sort_file") {
>     print "<p>Output also saved to:<br>"
>         . "  <tt>$sort_file</tt><p><hr>\n";
>
>     open INDEX, "<$sort_file"
>       or die "Can't open sorter file ($sort_file)";
>
>     Caseman::read_host_config();
>     my $mnt = $Caseman::vol2mnt{$Args::args{'vol'}};
>
>     my $str = "";
>     my $dir = "";
>     my $meta = "";
>
>     while (<INDEX>) {
>       next if ((/^<HTML><HEAD><TITLE>/i)
>               || (/^<BODY><H2>/i));
>
>       #match host filename
>       if (/^$mnt(.*)<BR>$/i) {
>         $dir = $1;
>         $str = "$mnt$dir" . "</a><BR>";
>       }
>
>       #match meta information
>       elsif (/^( ){2}(.*\s+Inode:\s+)(\d+)<BR>$/i) {
>         $meta = $3;
>         print "<a href=\"$::PROGNAME?mod=$::MOD_FILE&view=$File::CONT_FR&"
>             . "$Args::baseargs&meta=$meta&dir=$dir&recmode=1\"
target=\"file_content\">$str"
>             . "$1$2<a href=\"$::PROGNAME?mod=$::MOD_META&$Args::baseargs&"
>             . "view=$Meta::STATS&meta=$meta\"
target=\"file_content\">$meta</a><BR>\n"
>             . "<A
HREF=\"$::PROGNAME?mod=$::MOD_NOTES&view=$Notes::ENTER_FILE"
>             . "&$Args::baseargs&meta=$meta&dir=$dir\" TARGET=\"_blank\">"
>             . "Add Note </A><BR>\n";
>       }
>
>       #match file type information
>       elsif (/^( ){2}.*<BR>$/i) {
>         $str .= $_;
>       }
>       else {
>         print $_;
>         $str = "";
>       }
>     }
>   }
>   return 0;
> }
>
> # View thumbnails of graphic images if sorter was executed with this option
> sub view_thumbs {
```

```
>   my $MAX_THUMBS = 100;
>   Print::print_html_header("Sorted Thumbnails");
>
>   my $graphics_dir = get_sorter_graphics_dir();
>
>   if (-e "$graphics_dir/images.html") {
>     open IMAGES, "<$graphics_dir/images.html"
>       or die "Can't open images file ($graphics_dir/images.html)";
>
>     Caseman::read_host_config();
>     my $mnt = $Caseman::vol2mnt{$Args::args{'vol'}};
>     my $dir;
>     my $dir_lnk;
>     my $meta;
>
>     my $skip = ($Args::args{'page'} -1) * $MAX_THUMBS;
>     my $count = 0;
>     my $row = 1;
>
>     print "<CENTER><H2>Image Thumbnails - Page $Args::args{'page'}</H2>\n"
>         . "<P><TABLE WIDTH=630 CELLSPACING=5 CELLPADDING=0 BORDER=0>\n"
>         . "<TR><TD></TD><TD ALIGN=CENTER>A</TD><TD ALIGN=CENTER>B</TD>"
>         . "<TD ALIGN=CENTER>C</TD><TD ALIGN=CENTER>D</TD></TR>\n";
>
>     while (<IMAGES>) {
>       next if ((/^<HTML><HEAD><TITLE>/i)
>             || (/^<BODY>/i)
>             || (/^<CENTER><H2>/));
>
>       last if ($count >= ($MAX_THUMBS * $Args::args{'page'}) );
>
>       #parse images file for filenames on the host
>       if (/^$mnt(.*)<BR>$/i) {
>         $dir = $1;
>         if ($dir =~ /^(&lt;)(.*)(&gt;)$/) {
>             $dir_lnk = Args::url_encode($2);
>         }
>         else {
>             $dir_lnk = Args::url_encode($dir);
>         }
>       }
>
>       #match meta information
>       elsif (/^( ){2}(.*\s+Inode:\s+)($::REG_META)<BR>$/i) {
>         if ($count < $skip) {
>             $count++;
>         }
>
>         else {
>           $meta = $3;
>
>           if ($count % 4 == 0) {
>             print "<TR>\n<TD>$row</TD>\n";
>             $row++;
>           }
>           $count++;
>           print "<TD WIDTH=150><A HREF=\"$::PROGNAME?mod=$::MOD_APPVIEW&"
>           . "view=$Appview::CELL_CONT&$Args::baseargs&meta=$meta&"
>           . "dir=$dir_lnk"
>           . "&cell_mode=2&recmode=1\"\" TARGET=_blank>"
>           . "<IMG
SRC=\"$::PROGNAME?mod=$::MOD_APPVIEW&view=$Appview::CELL_CONT&"
```

```
>           . "$Args::baseargs&meta=$meta&dir=$dir_lnk"
>           . "&cell_mode=2&recmode=1\" "
>           . "WIDTH=150 HEIGHT=150";
>           $dir =~ /\/?([^\/]*)$/;
>           print " ALT=\"$1\"><BR>$1</A><BR>"
>           . "<A HREF=\"$::PROGNAME?mod=$::MOD_NOTES&view=$Notes::ENTER_FILE"
>           . "&$Args::baseargs&meta=$meta&dir=$dir\" TARGET=\"_blank\">"
>           . "Add Note </A><BR><BR></TD>\n";
>
>           if ($count % 4 == 0) {
>             print "</TR>\n";
>           }
>         }
>       }
>     }
>   }
>
>   Print::print_html_footer();
>   return 0;
> }
>
406a620,628
> # Blank Page for results of file categories
> sub cat_blank {
>     Print::print_html_header("");
>     print "<center><h3>File Type Sorting Results</h3>\n"
>       . "The contents of the files can be viewed here by clicking on their
links.";
>
>     Print::print_html_footer();
>     return 0;
> }
```

C. ARGS.PM

```
2c2
< # $Date: 2005/09/18 22:49:44 $
---
> # $Date: 2005/09/18 22:49:44 $
7a8
> # Regis Friend Cassidy [rfriendcassidy@gmail.com]
626a628,663
> # bookmark name
> ################################
> sub check_bmark {
>     if ( $Args::args{'bookmarknew'} ne "" )
>     {
>       unless ($Args::args{'bookmarknew'} =~ /^$::REG_BMARK$/) {
>         Print::print_check_err(
>             "Invalid bookmark name. You used special characters that are not
allowed");
>         return 1;
>       }
>       Notes::bookmark_add($Args::args{'bookmarknew'});
>     }
>     return 0;
> }
>
> sub get_bmark {
>     if ( (exists $Args::args{'bookmarknew'}) && ($Args::args{'bookmarknew'}
ne "")
>         && ($Args::args{'bookmarknew'} =~ /^($::REG_BMARK)$/) )
>     {
>         return $1;
>     }
>     elsif ( (exists $Args::args{'bookmark'}) && ($Args::args{'bookmark'} ne
"")
>         && ($Args::args{'bookmark'} =~ /^($::REG_BMARK)$/) )
>     {
>         return $1;
>     }
>     elsif ( ((exists $Args::args{'bookmarknew'}) &&
($Args::args{'bookmarknew'} eq ""))
>             ||
>             ((exists $Args::args{'bookmark'}) && ($Args::args{'bookmark'} eq
""))  )
>     {
>         return "";
>     }
>     Print::print_err("Invalid bookmark name");
> }
>
> ################################
674a712,764
>
> ########################################
> # st_day: start day of the month (0-31)
> # Used for Timeline analysis
> ########################################
> sub check_st_day {
>    if (   (exists $Args::args{'st_day'})
>        && ($Args::args{'st_day'} =~ /^(\d\d?)$/))
>    {
```

```
>        if (($1 < 1) || ($1 > 31)) {
>            print("Invalid start day\n");
>            return 1;
>        }
>    }
>    else {
>        print("Invalid start day\n");
>        return 1;
>    }
> }
>
> sub get_st_day {
>     if ($Args::args{'st_day'} =~ /^(\d\d?)$/) {
>        return $1;
>     }
>     Print::print_err("Invalid Day");
> }
>
> ######################################
> # end_day: end day of the month (0-31)
> # Used for Timeline analysis
> ######################################
> sub check_end_day {
>    if (   (exists $Args::args{'end_day'})
>        && ($Args::args{'end_day'} =~ /^(\d\d?)$/))
>    {
>        if (($1 < 1) || ($1 > 31)) {
>            print("Invalid end day\n");
>            return 1;
>        }
>    }
>    else {
>        print("Invalid end day\n");
>        return 1;
>    }
> }
>
> sub get_end_day {
>     if ($Args::args{'end_day'} =~ /^(\d\d?)$/) {
>        return $1;
>     }
>     Print::print_err("Invalid Day");
> }
>
```

D. CASEMAN.PM

```
2c2
< # $Date: 2005/09/18 22:49:44 $
---
> # $Date: 2005/09/18 22:49:44 $
7a8
> # Regis Friend Cassidy [rfriendcassidy@gmail.com]
48c49
<
---
> $Caseman::CASE_REPORTS  = 6;
50c51,56
< my $CASE_MAX = 5;
---
> my $CASE_MAX = 6;
>
> # Ivestigator Views
> $Caseman::INVEST_ADD   = 7;
> $Caseman::INVEST_ADD_DOIT = 8;
> my $INVEST_MAX = 8;
53,57c59,63
< $Caseman::HOST_ADD      = 7;
< $Caseman::HOST_ADD_DOIT = 8;
< $Caseman::HOST_OPEN     = 9;
< $Caseman::HOST_OPEN_LOG = 10;
< $Caseman::HOST_DETAILS  = 11;
---
> $Caseman::HOST_ADD      = 9;
> $Caseman::HOST_ADD_DOIT = 10;
> $Caseman::HOST_OPEN     = 11;
> $Caseman::HOST_OPEN_LOG = 12;
> $Caseman::HOST_DETAILS  = 13;
60c66
< my $HOST_MAX = 11;
---
> my $HOST_MAX = 13;
63,72c69,78
< $Caseman::IMG_ADD      = 13;
< $Caseman::IMG_ADD_PREP = 14;
< $Caseman::IMG_ADD_DOIT = 15;
< $Caseman::VOL_OPEN     = 16;
< $Caseman::VOL_OPEN_LOG = 17;
< $Caseman::VOL_DETAILS  = 18;
< $Caseman::IMG_DEL      = 19;
< $Caseman::VOL_MAKESTR  = 20;
< $Caseman::VOL_MAKEDLS  = 21;
< my $IMG_MAX = 21;
---
> $Caseman::IMG_ADD      = 14;
> $Caseman::IMG_ADD_PREP = 15;
> $Caseman::IMG_ADD_DOIT = 16;
> $Caseman::VOL_OPEN     = 17;
> $Caseman::VOL_OPEN_LOG = 18;
> $Caseman::VOL_DETAILS  = 19;
> $Caseman::IMG_DEL      = 20;
> $Caseman::VOL_MAKESTR  = 21;
> $Caseman::VOL_MAKEDLS  = 22;
> my $IMG_MAX = 22;
142a149,151
```

```
>      elsif ($view == $Caseman::CASE_REPORTS) {
>          return case_reports();
>         }
148a158,167
>     #Investigator Functions
>     if ($view <= $INVEST_MAX) {
>       if ($view == $Caseman::INVEST_ADD) {
>      return invest_add();
>       }
>       elsif ($view == $Caseman::INVEST_ADD_DOIT) {
>      return invest_add_doit();
>       }
>     }
>
328,331c347,350
<         next if ((/^\#/) || (/^\s+$/));
<         s/^\s+//;
<         s/\s+$//;
<         $Caseman::cvals{$1} = $2 if (/^(\S+)\s+(.*)$/);
---
>       next if ((/^\#/) || (/^\s+$/));
>       s/^\s+//;
>       s/\s+$//;
>       $Caseman::cvals{$1} = $2 if (/^(\S+)\s+(.*)$/);
359c378
<     1.  <b>Case Name:</b> The name of this investigation.  It can contain
only letters, numbers, and symbols.
---
>     1.  <b>Case Name:</b> The name of this investigation. (No spaces)
370,371c389,390
<   <td colspan=3 align=left>
<     2.  <b>Description:</b> An optional, one line description of this case.
---
>   <td colspan=3 align=left >
>     2.  <b>Description:</b> Optional description of this case
376c395
<   <td align=left colspan=2><input type=\"text\" name=\"desc\" size=32
maxlength=32></td>
---
>   <td align=left colspan=2><textarea name=\"desc\" cols=50></textarea></td>
382,384c401,412
<   <td colspan=3 align=left>
<     3.  <b>Investigator Names:</b> The optional names (with no spaces) of the
investigators for this case.
<   </td>
---
>    <td align=\"left\" colspan=\"3\">
>      3. <b>Forensic Investigator:</b> Additional Investigators can be added
later
>    </td>
> <tr>
>    <td>  </td>
>    <td align=\"right\">Name: </td>
>    <td align=\"left\" width=\"100%\"><input type=\"text\" name=\"inv\"></td>
> </tr>
> <tr>
>    <td>  </td>
>    <td align=\"right\">Company/Agency: </td>
>    <td align=\"left\"><input type=\"text\" name=\"comp\"></td>
387,389c415,417
<   <td> </td>
<   <td align=left><tt>a.</tt> <input type=\"text\" name=\"inv1\"></td>
```

```
<   <td align=left><tt>b.</tt> <input type=\"text\" name=\"inv2\"></td>
---
>    <td>  </td>
>    <td align=\"right\">Title: </td>
>    <td align=\"left\"><input type=\"text\" name=\"title\"></td>
392,394c420,422
<   <td> </td>
<   <td align=left><tt>c.</tt> <input type=\"text\" name=\"inv3\"></td>
<   <td align=left><tt>d.</tt> <input type=\"text\" name=\"inv4\"></td>
---
>    <td>  </td>
>    <td align=\"right\">Address: </td>
>    <td align=\"left\"><textarea name=\"addr\" cols=\"32\"></textarea></td>
397,399c425,427
<   <td> </td>
<   <td align=left><tt>e.</tt> <input type=\"text\" name=\"inv5\"></td>
<   <td align=left><tt>f.</tt> <input type=\"text\" name=\"inv6\"></td>
---
>    <td>  </td>
>    <td align=\"right\">Phone: </td>
>    <td align=\"left\"><input type=\"text\" name=\"phone\"></td>
402,404c430,432
<   <td> </td>
<   <td align=left><tt>g.</tt> <input type=\"text\" name=\"inv7\"></td>
<   <td align=left><tt>h.</tt> <input type=\"text\" name=\"inv8\"></td>
---
>    <td>  </td>
>    <td align=\"right\">Email: </td>
>    <td align=\"left\"><input type=\"text\" name=\"email\"</td>
407,409c435,438
<   <td> </td>
<   <td align=left><tt>i.</tt> <input type=\"text\" name=\"inv9\"></td>
<   <td align=left><tt>j.</tt> <input type=\"text\" name=\"inv10\"></td>
---
>    <td>  </td>
>    <td align=\"right\">Comments: </td>
>    <td align=\"left\"><textarea name=\"comm\" cols=\"32\"></textarea></td>
> <tr><td colspan=\"3\"> </td></tr>
410a440
>
441a472
>
444a476
>
465c497
<     Print::log_case_info("Case $case created");
---
>     Print::log_case_info("Case $case created by $Args::args{'inv'}");
477,481c509,511
<         Print::print_err("Invalid Description<br>\n"
<               . "Use the browser's back button to fix")
<           if ($Args::args{'desc'} =~ /\n/);
<
<         print CASE_CONFIG "desc $Args::args{'desc'}\n";
---
>         #replace newlines with html <br> tag
>         $Args::args{'desc'} =~ s/\r\n/<br>/g;
>      print CASE_CONFIG "desc $Args::args{'desc'}\n";
484,485c514,515
<     print CASE_CONFIG "data           $::DATADIR\n";
<     print CASE_CONFIG "log            $::LOGDIR\n";
---
```

```
>     print CASE_CONFIG "data    $::DATADIR\n";
>     print CASE_CONFIG "log     $::LOGDIR\n";
494,561c524,528
<     my @invs;
<     if (   (exists $Args::args{'inv1'})
<         && ($Args::args{'inv1'} ne "")
<         && ($Args::args{'inv1'} =~ /^\s*($::REG_INVESTIG)\s*$/o))
<     {
<         print INVES "$1\n";
<         push @invs, $1;
<     }
<     if (   (exists $Args::args{'inv2'})
<         && ($Args::args{'inv2'} ne "")
<         && ($Args::args{'inv2'} =~ /^\s*($::REG_INVESTIG)\s*$/o))
<     {
<         print INVES "$1\n";
<         push @invs, $1;
<     }
<     if (   (exists $Args::args{'inv3'})
<         && ($Args::args{'inv3'} ne "")
<         && ($Args::args{'inv3'} =~ /^\s*($::REG_INVESTIG)\s*$/o))
<     {
<         print INVES "$1\n";
<         push @invs, $1;
<     }
<     if (   (exists $Args::args{'inv4'})
<         && ($Args::args{'inv4'} ne "")
<         && ($Args::args{'inv4'} =~ /^\s*($::REG_INVESTIG)\s*$/o))
<     {
<         print INVES "$1\n";
<         push @invs, $1;
<     }
<     if (   (exists $Args::args{'inv5'})
<         && ($Args::args{'inv5'} ne "")
<         && ($Args::args{'inv5'} =~ /^\s*($::REG_INVESTIG)\s*$/o))
<     {
<         print INVES "$1\n";
<         push @invs, $1;
<     }
<     if (   (exists $Args::args{'inv6'})
<         && ($Args::args{'inv6'} ne "")
<         && ($Args::args{'inv6'} =~ /^\s*($::REG_INVESTIG)\s*$/o))
<     {
<         print INVES "$1\n";
<         push @invs, $1;
<     }
<     if (   (exists $Args::args{'inv7'})
<         && ($Args::args{'inv7'} ne "")
<         && ($Args::args{'inv7'} =~ /^\s*($::REG_INVESTIG)\s*$/o))
<     {
<         print INVES "$1\n";
<         push @invs, $1;
<     }
<     if (   (exists $Args::args{'inv8'})
<         && ($Args::args{'inv8'} ne "")
<         && ($Args::args{'inv8'} =~ /^\s*($::REG_INVESTIG)\s*$/o))
<     {
<         print INVES "$1\n";
<         push @invs, $1;
<     }
<     if (   (exists $Args::args{'inv9'})
<         && ($Args::args{'inv9'} ne "")
```

```
<         && ($Args::args{'inv9'} =~ /^\s*($::REG_INVESTIG)\s*$/o))
<     {
<         print INVES "$1\n";
<         push @invs, $1;
<     }
<     if (   (exists $Args::args{'inv10'})
<         && ($Args::args{'inv10'} ne "")
<         && ($Args::args{'inv10'} =~ /^\s*($::REG_INVESTIG)\s*$/o))
<     {
---
>     if (   (exists $Args::args{'inv'})
>         && ($Args::args{'inv'} ne "")
>         && ($Args::args{'inv'} =~ /^\s*($::REG_INVESTIG)$/o))
>       {
>      print "Investigator <tt>$Args::args{'inv'}</tt> added to
<tt>$iname</tt><br>\n";
563c530,545
<         push @invs, $1;
---
>     if ( $Args::args{'comp'} ne "" || $Args::args{'title'} ne "" ||
$Args::args{'addr'} ne ""
>              || $Args::args{'phone'} ne "" || $Args::args{'email'} ne "" ||
$Args::args{'comm'} ne "")
>       {
>         open IINFO, ">$::case_dir/$1.aut" or die "Can't open investigator's
info file: $::case_dir/$1.aut";
>         print IINFO "comp $Args::args{'comp'}\n" if ( $Args::args{'comp'} ne
"");
>         print IINFO "title $Args::args{'title'}\n" if ( $Args::args{'title'}
ne "");
>         print IINFO "addr $Args::args{'addr'}\n" if ( $Args::args{'addr'} ne
"");
>         print IINFO "phone $Args::args{'phone'}\n" if ( $Args::args{'phone'}
ne "");
>         print IINFO "email $Args::args{'email'}\n" if ( $Args::args{'email'}
ne "");
>         print IINFO "comm $Args::args{'comm'}\n" if ( $Args::args{'comm'} ne
"");
>       }
>       }
>     else {
>       print "<br>An investigator was not added for this case. Either none was
entered"
>           . " or a name was incorrectly entered. You can always add an
investigator later"
>           . " from the Host Gallery Page.<br><br>\n";
568c550
<     Print::log_session_info("Case $case created");
---
>     Print::log_session_info("Case $case created by $Args::args{'inv'}");
577,578c559,560
<     if (scalar @invs == 0) {
<         print "<option hiddden name=\"inv\" value=\"unknown\">\n";
---
>     if (! exists $Args::args{'inv'}) {
>         print "<input type=\"hidden\" name=\"inv\" value=\"unknown\">\n";
581,587c563
<         print "<br><br>Please select your name from the list: "
<           . "<select name=\"inv\" size=\"1\">\n";
<
<         foreach $i (@invs) {
<             print "<option value=\"$i\">$i</option>\n";
```

```
<         }
<         print "</select>\n";
---
>       print "<input type=\"hidden\" name=\"inv\"
value=\"$Args::args{'inv'}\">\n";
643,644c619,620
<             print "<tr><td align=\"left\">"
<               . "<input type=\"radio\" name=\"case\" value=$c";
---
>             print "<tr valign=\"top\"><td align=\"left\">"
>               . "<input type=\"radio\" name=\"case\" value=\"$c\"";
652,653c628,644
<
<             print "<td>$Caseman::cvals{'desc'}</td>"
---
>
>             #make preview of descritpions 32 chars long
>             my $desc;
>             if (length($Caseman::cvals{'desc'}) > 32 ) {
>                 #check and see if 32 chars breaks up a <br> tag
>                 if ($Caseman::cvals{'desc'} =~ /^(.{29,31})<br>/) {
>                     $desc = $1 . " ...";
>                 }
>                 else {
>                     $desc = substr $Caseman::cvals{'desc'}, 0, 32;
>                     $desc .= " ...";
>                 }
>             }
>             else {
>                 $desc = $Caseman::cvals{'desc'};
>             }
>          print "<td>$desc</td>"
657,659c648,649
<               . "details</a></td>"
<               . "</tr>\n";
<         }
---
>               . "details</a></td>";
>            }
661,662c651,653
<     }
<
---
>
>         }
>
691c682,684
<
---
>
>
>     # Print Help Button
693c686
<       . "<td> </td>\n"
---
>       . "<td> </td>"
714a708
>
738c732
<       "  <tr><td align=\"right\"><b>Description:</b></td>"
---
>       "  <tr><td align=\"right\" valign=\"top\"><b>Description:</b></td>"
```

```
1017a1012,1016
>         $Caseman::host_created = "";
>         $Caseman::host_inv   = "";
>         $Caseman::vol2tadded = "";
>         $Caseman::vol2iadded = "";
>
1022d1020
<
1024c1022,1023
<   restart:
---
>
>    restart:
1030a1030,1031
>     %Caseman::vol2itype = ();
>     %Caseman::vol2splitname = ();
1037a1039
>     %Caseman::vol2part  = ();
1040,1041c1042,1045
<     $Caseman::vol2path  = ();
<     $Caseman::vol2sname = ();
---
>     %Caseman::vol2path  = ();
>     %Caseman::vol2sname = ();
>     %Caseman::vol2tadded = ();
>     %Caseman::vol2iadded = ();
1047a1052,1053
>     $Caseman::host_created = "";
>     $Caseman::host_inv   = "";
1050c1056,1058
<         next if ((/^\#/) || (/^\s+$/));
---
>      unless (/^\#\sCreated/) {
>            next if ((/^\#/) || (/^\s+$/));
>         }
1063c1071
<             goto restart;
---
>             #goto restart;
1066c1074
<
/^image\s+($::REG_INAME)\s+($::REG_IMGTYPE)\s+($::REG_IMG_CONFIG)$/o
---
>
/^image\s+($::REG_INAME)\s+($::REG_IMGTYPE)\s+($::REG_IMG_CONFIG)/o
1068c1076
<         {
---
>          {
1071a1080,1086
>             my $tadded = "";
>             my $iadded = "";
>
>             if ( /'(.*)'\s+($::REG_INVESTIG)$/o ) {
>             $tadded = $1;            #time image added
>                $iadded = $2;            #investigator that added image
>             }
1085a1101,1102
>             my $name = $1 if ($i =~ /\/($::REG_FILE)$/);
>             $Caseman::vol2splitname{$me} .= ", " . $name;
1090a1108
>              $Caseman::vol2splitname{$me} = $Caseman::vol2sname{$me};
```

```
1099c1117,1119
<             }
---
>                 $Caseman::vol2tadded{$me} = $tadded;
>                 $Caseman::vol2iadded{$me} = $iadded;
>             }
1140a1161
>             push @{ $Caseman::vol2part{$par} } , $me;
1170a1192
>          push @{ $Caseman::vol2part{$par} }, $me;
1176c1198
<         elsif (/^strings\s+($::REG_VNAME)\s+($::REG_VNAME)\s+($::REG_IMG)$/o)
{
---
>         elsif (/^strings\s+($::REG_VNAME)\s+($::REG_VNAME)\s+($::REG_IMG)/o)
{
1179a1202,1208
>             my $tadded = "";
>             my $iadded = "";
>
>             if ( /'(.*)'\s+($::REG_INVESTIG)$/o ) {
>             $tadded = $1;            #time image added
>                $iadded = $2;            #investigator that added image
>             }
1206c1235,1236
<
---
>          $Caseman::vol2tadded{$me} = $tadded;
>          $Caseman::vol2iadded{$me} = $iadded;
1212c1242
<         elsif
(/^unistrings\s+($::REG_VNAME)\s+($::REG_VNAME)\s+($::REG_IMG)$/o)
---
>         elsif
(/^unistrings\s+($::REG_VNAME)\s+($::REG_VNAME)\s+($::REG_IMG)/o)
1216a1247,1253
>             my $tadded = "";
>             my $iadded = "";
>
>             if ( /'(.*)'\s+($::REG_INVESTIG)$/o ) {
>             $tadded = $1;            #time image added
>                $iadded = $2;            #investigator that added image
>             }
1242a1280,1281
>          $Caseman::vol2tadded{$me} = $tadded;
>          $Caseman::vol2iadded{$me} = $iadded;
1247c1286
<         elsif (/^dls\s+($::REG_VNAME)\s+($::REG_VNAME)\s+($::REG_IMG)$/o) {
---
>         elsif (/^dls\s+($::REG_VNAME)\s+($::REG_VNAME)\s+($::REG_IMG)/o) {
1250a1290,1296
>             my $tadded = "";
>             my $iadded = "";
>
>             if ( /'(.*)'\s+($::REG_INVESTIG)$/o ) {
>             $tadded = $1;            #time image added
>                $iadded = $2;            #investigator that added image
>             }
1280a1327,1328
>          $Caseman::vol2tadded{$me} = $tadded;
>          $Caseman::vol2iadded{$me} = $iadded;
1285c1333
```

```
<         elsif (/^body\s+($::REG_VNAME)\s+($::REG_IMG)$/o) {
---
>         elsif (/^body\s+($::REG_VNAME)\s+($::REG_IMG)/o) {
1287a1336,1343
>             my $tadded = "";
>             my $iadded = "";
>
>             if ( /'(.*)'\s+($::REG_INVESTIG)$/o ) {
>                $tadded = $1;            #time image added
>                $iadded = $2;            #investigator that added image
>             }
>
1304a1361,1363
>             $Caseman::vol2tadded{$me} = $tadded;
>             $Caseman::vol2iadded{$me} = $iadded;
>
1308c1367
<         elsif (/^timeline\s+($::REG_VNAME)\s+($::REG_IMG)$/o) {
---
>         elsif (/^timeline\s+($::REG_VNAME)\s+($::REG_IMG)/o) {
1310a1370,1377
>             my $tadded = "";
>             my $iadded = "";
>
>             if ( /'(.*)'\s+($::REG_INVESTIG)$/o ) {
>                $tadded = $1;            #time image added
>                $iadded = $2;            #investigator that added image
>             }
>
1329a1397,1398
>             $Caseman::vol2tadded{$me} = $tadded;
>             $Caseman::vol2iadded{$me} = $iadded;
1332a1402,1413
>         # created time
>         elsif (/^\#\sCreated:\s+(.*)$/o) {  #Old Autopsy host.aut format
>            $Caseman::host_created = "$1";
>         }
>         elsif (/^created\s+(.*)$/o) {
>            $Caseman::host_created = "$1";
>         }
>
>         elsif (/^invest\s+($::REG_INVESTIG)$/o) {
>            $Caseman::host_inv = "$1";
>         }
>
1395c1476,1477
<     print WRITE "$type  $id  $other\n";
---
>
>     print WRITE "$type  $id  $other '" . localtime() . "'
$Args::args{'inv'}\n";
1397c1479
<     Print::log_host_info("Image added: $type $id $other");
---
>     Print::log_host_info("Image added: $type $id $other by
$Args::args{'inv'}");
1438,1439c1520,1527
<     print WRITE "$type  vol" . $maxcnt . "  $other\n";
<     Print::log_host_info("Volume added: $type  vol" . $maxcnt . " $other");
---
>     print WRITE "$type  vol" . $maxcnt . "  $other";
>     if ( ($type ne "disk") && ($type ne "part") ) {
```

```
>      print WRITE  " '" . localtime() . "' $Args::args{'inv'}\n";
>     }
>     else {
>         print WRITE "\n";
>     }
>     Print::log_host_info("Volume added: $type  vol" . $maxcnt . " $other" . "
by $Args::args{'inv'}");
1535a1624,1742
> sub invest_add {
>   Print::print_html_header("Add a New Investigator");
>   print <<EOF;
>     <center>
>     <b>Add a New Investigator</b>
>     <form action=\"$::PROGNAME\" method=\"get\">
>       <input type=\"hidden\" name=\"mod\" value=\"$::MOD_CASEMAN\">
>       <input type=\"hidden\" name=\"view\"
value=\"$Caseman::INVEST_ADD_DOIT\">
>       <input type=\"hidden\" name=\"case\" value=\"$Args::args{'case'}\">
>       <table width=\"600\" background=\"$::YEL_PIX\" cellspacing=\"0\"
>           cellpadding=\"2\" border=0>
>         <tr>
>            <td align=\"right\"><br>Name: </td>
>            <td align=\"left\" width=\"100%\"><br><input type=\"text\"
name=\"inv\"></td>
>         </tr>
>         <tr>
>            <td align=\"right\">Company/Agency: </td>
>            <td align=\"left\"><input type=\"text\" name=\"comp\"></td>
>         </tr>
>         <tr>
>            <td align=\"right\">Title: </td>
>            <td align=\"left\"><input type=\"text\" name=\"title\"></td>
>         </tr>
>         <tr>
>            <td align=\"right\">Address: </td>
>            <td align=\"left\"><textarea name=\"addr\"
cols=\"32\"></textarea></td>
>         </tr>
>         <tr>
>            <td align=\"right\">Phone: </td>
>            <td align=\"left\"><input type=\"text\" name=\"phone\"></td>
>         </tr>
>         <tr>
>            <td align=\"right\">Email: </td>
>            <td align=\"left\"><input type=\"text\" name=\"email\"</td>
>         </tr>
>         <tr>
>            <td align=\"right\">Comments: </td>
>            <td align=\"left\"><textarea name=\"comm\"
cols=\"32\"></textarea></td>
>         <tr><td> </td></tr>
>       </table>
>       <br>
>     <table><tr><td align=\"right\"><input type=\"image\"
src=\"pict/menu_b_ok.jpg\" "width=167 height=20 alt=\"Ok\" border=0>
>     </form></td>
>     <td align=\"left\"><form action=\"$::PROGNAME\" method=\"get\">
>       <input type=\"hidden\" name=\"mod\" value=\"$::MOD_CASEMAN\">
>       <input type=\"hidden\" name=\"view\" value=\"$Caseman::HOST_OPEN\">
>       <input type=\"hidden\" name=\"case\" value=\"$Args::args{'case'}\">
>       <input type=\"image\" src=\"pict/menu_b_cancel.jpg\" alt=\"Cancel\"
width=\"167\" height=20 border=0>
```

```
>     </form></td></tr></table>
>     </center>
>
> EOF
>   Print::print_html_footer();
>   return;
> }
> sub invest_add_doit {
>   Print::print_html_header("Adding Investigator: $Args::args{'inv'}");
>   my $inv = $Args::args{'inv'};
>
>   print "<h3>Adding Investigator: <tt>$inv</tt></h3>\n";
>
>   #Check if Investigator already exists
>   my $dup = 0;
>   my @invs = read_invest();
>   foreach my $i (@invs) {
>     $dup = 1 if ($inv eq $i);
>   }
>
>   if (! $dup) {
>     Print::log_case_info("Investigator $inv added to case");
>
>     my $iname = investig_fname();
>     open INVES, ">>$iname" or die "Can't open investigators file: $iname";
>
>     if (   (exists $Args::args{'inv'})
>         && ($Args::args{'inv'} ne "")
>         && ($Args::args{'inv'} =~ /^\s*($::REG_INVESTIG)$/o))
>       {
>      print "Investigator <tt>$Args::args{'inv'}</tt> added to
<tt>$iname</tt><br>\n";
>         print INVES "$1\n";
>      if ( $Args::args{'comp'} ne "" || $Args::args{'title'} ne "" ||
$Args::args{'addr'} ne ""
>               || $Args::args{'phone'} ne "" || $Args::args{'email'} ne "" ||
$Args::args{'comm'} ne "")
>        {
>          open IINFO, ">$::case_dir/$1.aut" or die "Can't open investigator's
info file: $::case_dir/$1.aut";
>          print IINFO "comp $Args::args{'comp'}\n" if ( $Args::args{'comp'} ne
"");
>          print IINFO "title $Args::args{'title'}\n" if ( $Args::args{'title'}
ne "");
>          print IINFO "addr $Args::args{'addr'}\n" if ( $Args::args{'addr'} ne
"");
>          print IINFO "phone $Args::args{'phone'}\n" if ( $Args::args{'phone'}
ne "");
>          print IINFO "email $Args::args{'email'}\n" if ( $Args::args{'email'}
ne "");
>          print IINFO "comm $Args::args{'comm'}\n" if ( $Args::args{'comm'} ne
"");
>        }
>       }
>     else {
>       print "<br>An investigator was not added for this case. Either none was
entered"
>           . " or a name was incorrectly entered. Please try to add an
investigator again"
>           . " from the Host Gallery Page.<br><br>\n";
>     }
>
```

```
>     close(INVES);
>   }
>
>   #investigator already exists
>   else {
>     $inv = "";
>     print "This investigator was not added because they already
exist\n<br><br>\n";
>   }
>
>   print "<form action=\"$::PROGNAME\" method=\"get\">\n"
>       . "<input type=\"hidden\" name=\"mod\" value=\"$::MOD_CASEMAN\">\n"
>       . "<input type=\"hidden\" name=\"view\"
value=\"$Caseman::HOST_OPEN\">\n"
>       . "<input type=\"hidden\" name=\"case\"
value=\"$Args::args{'case'}\">\n"
>       . "<input type=\"hidden\" name=\"inv\" value=\"$inv\">\n<br><br>"
>       . "<input type=\"image\" src=\"pict/menu_b_ok.jpg\" alt=\"Back to Host
Gallery\" "
>       . "height=20 border=\"0\"></form>\n";
>
>   Print::print_html_footer();
>   return;
> }
>
1536a1744,1774
>     unless ((exists $Args::args{'inv'}) && ($Args::args{'inv'} ne "")) {
>         my @invs = read_invest();
>         if (scalar @invs == 0) {
>             $Args::args{'inv'} = $Args::enc_args{'inv'} = 'unknown';
>         }
>         else {
>             Print::print_html_header("Missing Investigator");
>             print "<br>An investigator must be selected<p>\n"
>               . "<form action=\"$::PROGNAME\" method=\"get\">\n"
>               . "<input type=\"hidden\" name=\"mod\"
value=\"$::MOD_CASEMAN\">\n"
>               . "<input type=\"hidden\" name=\"view\"
value=\"$Caseman::HOST_ADD\">\n"
>               . Args::make_hidden();
>
>             print "Select one of the following:";
>             print "<select name=\"inv\" size=\"1\">\n";
>
>             print "<option value=\"\" selected>Select One" . "</option>\n";
>
>             foreach my $i (@invs) {
>                 print "<option value=\"$i\">$i</option>\n";
>             }
>             print "</select><p>\n"
>               . "<input type=\"image\" src=\"pict/but_ok.jpg\" alt=\"Ok\" "
>               . "width=43 height=20 border=\"0\">\n"
>               . "</form>\n";
>
>             Print::print_html_footer();
>             return 0;
>         }
>     }
>
1561c1799
< "<tr><td align=\"left\" colspan=2>2.  <b>Description:</b>  An optional one-
line description or note about this computer.</td></tr>"
```

```
---
> "<tr><td align=\"left\" colspan=2>2.  <b>Description:</b>  An optional
description or note about this computer.</td></tr>"
1564c1802
<       . "<input type=\"text\" name=\"desc\" size=32
maxlength=32></td></tr>\n"
---
>       . "<textarea name=\"desc\" cols=50></textarea></td></tr>\n"
1715c1953
<     Print::log_case_info("Host $Args::args{'host'} added to case");
---
>     Print::log_case_info("Host $Args::args{'host'} added to case by
$Args::args{'inv'}");
1758c1996
<         "Host $Args::args{'host'} added to case $Args::args{'case'}");
---
>         "Host $Args::args{'host'} added to case $Args::args{'case'} by
$Args::args{'inv'}");
1765,1767c2003,2007
<       . "# Case: $Args::args{'case'}        Host: $Args::args{'host'}\n"
<       . "# Created: "
<       . localtime() . "\n\n";
---
>       . "# Case: $Args::args{'case'}        Host: $Args::args{'host'}\n\n"
>       . "created "
>       . localtime() . "\n"
>       #print investigator adding host to case
>       . "invest $Args::args{'inv'}\n";
1770,1773c2010,2011
<         Print::print_err("Invalid Description<br>\n"
<               . "Use the browser's back button to fix")
<           if ($Args::args{'desc'} =~ /\n/);
<
---
>         #replace newlines with html <br> tag
>         $Args::args{'desc'} =~ s/\r\n/<br>/g;
1854a2093,2096
>     print "<form action=\"$::PROGNAME\" method=\"get\">\n"
>         . "<input type=\"hidden\" name=\"mod\" value=\"$::MOD_CASEMAN\">\n"
>         . Args::make_hidden();
>
1866,1873c2108,2111
<           . "background=\"$::YEL_PIX\" border=0>\n";
<
<         print "<form action=\"$::PROGNAME\" method=\"get\">\n"
<           . "<input type=\"hidden\" name=\"mod\" value=\"$::MOD_CASEMAN\">\n"
<           . "<input type=\"hidden\" name=\"view\"
value=\"$Caseman::HOST_OPEN_LOG\">\n"
<           . Args::make_hidden()
<           . "<tr><th>Name</th>"
<           . "<th>Description</th><th> </th></tr>\n";
---
>             . "background=\"$::YEL_PIX\" border=0>\n"
>             . "<tr><td colspan=\"3\" align=\"center\">"
>          . "<tr align=\"left\"><th>Name</th>"
>             . "<th>Description</th><th> </th></tr>\n";
1900a2139,2149
>            #make preview of descritpions 32 chars long
>             if (length($desc) > 32 ) {
>                 #check and see if 32 chars breaks up a <br> tag
>                 if ($desc =~ /^(.{29,31})<br>/) {
>                     $desc = $1 . " ...";
```

```
>                 }
>                 else {
>                     $desc = substr $desc, 0, 32;
>                     $desc .= " ...";
>                 }
>             }
1909a2159,2160
>     }
>         print "<table border=\"0\"><tr align=\"center\" valign=\"middle\"><td
nowrap>";
1913c2164
<             print "<input type=\"hidden\" name=\"inv\" value=\"unknown\">\n";
---
>            print "<input type=\"hidden\" name=\"inv\" value=\"unknown\">\n";
1915,1920d2165
<         else {
<             print "<br>Investigator (for reports only): ";
<             my $cur_inv = "";
<             $cur_inv = $Args::args{'inv'} if (exists $Args::args{'inv'});
<
<             print "<select name=\"inv\" size=\"1\">\n";
1922,1933c2167,2191
<             if (($cur_inv eq "") && (scalar @invs != 1)) {
<                 print "<option value=\"\" selected>Select One" .
"</option>\n";
<             }
<             foreach my $i (@invs) {
<                 print "<option value=\"$i\"";
<                 print " selected" if ($cur_inv eq $i);
<                 print ">$i</option>\n";
<             }
<             print "</select>\n";
<         }
<     }
<     print "<br><br><table width=\"600\" cellspacing=\"0\"
cellpadding=\"2\">\n"
---
>         else {
>            print "<br>Investigator: ";
>            my $cur_inv = "";
>            $cur_inv = $Args::args{'inv'} if (exists $Args::args{'inv'} &&
$Args::args{'inv'} ne "");
>
>            print "<select name=\"inv\" size=\"1\">\n";
>
>            if (scalar @invs != 1) {
>               print "<option value=\"\" selected>Select One" . "</option>\n";
>            }
>            #if only one investigator automatically choose them
>            else {
>               $Args::args{'inv'} = $invs[0];
>            }
>
>            foreach my $i (@invs) {
>               print "<option value=\"$i\" ";
>               print " selected" if ($cur_inv eq $i);
>               print ">$i</option>\n";
>            }
>            print "</select>    ";
>         }
```

```
>         print "<a
href=\"$::PROGNAME?mod=$::MOD_CASEMAN&view=$Caseman::INVEST_ADD&case=$Args::arg
s{'case'}\">"
>             . "Add Investigator<br><br></a></td></tr>"
>             . "<tr align=\"center\"><td><table width=\"600\"
cellspacing=\"0\" cellpadding=\"2\">\n"
1942,1943c2200,2201
<           . "alt=\"Ok\" width=\"167\" height=20 border=0>\n"
<           . "</form>\n</td>\n";
---
>           . "alt=\"Ok\" width=\"167\" height=20 border=0 name=\"view\"
value=\"$Caseman::HOST_OPEN_LOG\">\n"
>           . "</td>\n";
1948,1951d2205
<       . "<form action=\"$::PROGNAME\" method=\"get\">\n"
<       . "<input type=\"hidden\" name=\"mod\" value=\"$::MOD_CASEMAN\">\n"
<       . "<input type=\"hidden\" name=\"view\"
value=\"$Caseman::HOST_ADD\">\n"
<       . "<input type=\"hidden\" name=\"case\"
value=\"$Args::args{'case'}\">\n"
1953,1955c2207,2208
<       . "alt=\"Add Host\" width=\"176\" height=20 border=0>\n"
<       . "</form></td>"
<       .
---
>       . "alt=\"Add Host\" width=\"176\" height=20 border=0 name=\"view\"
value=\"$Caseman::HOST_ADD\">\n"
>       . "</td>"
1957,1961c2210,2211
<       # Close Button
<       "<td align=center>"
<       . "<form action=\"$::PROGNAME\" method=\"get\">\n"
<       . "<input type=\"hidden\" name=\"mod\" value=\"$::MOD_CASEMAN\">\n"
<       . "<input type=\"hidden\" name=\"view\"
value=\"$Caseman::CASE_OPEN\">\n"
---
>     # Close Button
>       . "<td align=\"center\">"
1963,1964c2213,2217
<       . "alt=\"Close Case\" width=\"176\" height=20 border=0>\n"
<       . "</form></td></tr></table>\n";
---
>       . "alt=\"Close Case\" width=\"176\" height=20 border=0 name=\"view\"
value=\"$Caseman::CASE_OPEN\">\n"
>       . "</td></tr></table>\n";
>
>     print "<table width=\"400\"  cellspacing=\"0\" cellpadding=\"2\"
border=\"0\">\n"
>       . "<tr><td align=\"center\" width=\"50%\">"
1966,1968c2219,2225
<     print "<table width=\"600\"  cellspacing=\"0\" cellpadding=\"2\">\n"
<       . "<tr><td> </td>"
<       . "<td align=center><a href=\"$::HELP_URL\" "
---
>     # Case Report Button
>       . "<input type=\"image\" src=\"pict/menu_b_report.jpg\" alt=\"Case
Report\" "
>       . "border=0 name=\"mod\" value=\"$::MOD_REPORTS\">\n"
>       . "</form></td>\n"
>
>     # Help Button
>       . "<td align=center width=\"50%\"><a href=\"$::HELP_URL\" "
```

```
1972,1973c2229,2230
<       . "</a></td><td> </td></tr>\n"
<       . "</table>\n";
---
>       . "</a></td></tr>\n"
>       . "</table>\n</td></tr></table>\n";
2043c2300
<       "<tr><td align=\"right\"><b>Description:</b></td>"
---
>       "<tr><td align=\"right\" valign=\"top\"><b>Description:</b></td>"
2112c2369,2372
<     return "$::case_dir" . "investigators.txt";
---
>     Args::check_case();
>     $::case_dir = "$::LOCKDIR/" . Args::get_case() . "/";
>     $::case_dir =~ s/\/\//\//g;
>     return "$::case_dir" . "investigators.txt";
2293a2554
>           . "<input type=\"hidden\" name=\"inv\"
value=\"$Args::args{'inv'}\">\n"
3410c3671
<
---
>
3414d3674
<
3417c3677,3678
<
---
>     my $img = $Caseman::vol2par{$vol};
>
3430,3432c3691,3702
<       . "<td align=\"left\"><tt>$vol</tt></td></tr>\n"
<       . "<tr><td align=\"right\" width=\"300\"><b>Parent Volume Id:</b></td>"
<       . "<td align=\"left\"><tt>$Caseman::vol2par{$vol}</tt></td></tr>\n"
---
>       . "<td align=\"left\"><tt>$vol</tt></td></tr>\n";
>      unless ($Caseman::vol2tadded{$img} eq "") {
>         print "<tr><td align=\"right\" width=\"300\"><b>Time Added:</b></td>"
>             . "<td
align=\"left\"><tt>$Caseman::vol2tadded{$img}</tt></td></tr>\n";
>      }
>      unless ($Caseman::vol2iadded{$img} eq "") {
>         print "<tr><td align=\"right\" width=\"300\"><b>Added by:</b></td>"
>             . "<td
align=\"left\"><tt>$Caseman::vol2iadded{$img}</tt></td></tr>\n";
>      }
>
>      print "<tr><td align=\"right\" width=\"300\"><b>Parent Volume
Id:</b></td>"
>          . "<td align=\"left\"><tt>$Caseman::vol2par{$vol}</tt></td></tr>\n"
3752c4022
<                 . "$::DATA_DIR/"
---
>                 . "$::DATADIR/"
3976c4246
<             . "$::DATA_DIR/"
---
>             . "$::DATADIR/"
```

E. EXEC.PM

```
2c2
< # $Date: 2005/09/18 22:49:44 $
---
> # $Date: 2005/09/18 22:49:44 $
41a42,43
>     #untaint for case reports
>     $cmd = $1 if ($cmd =~ /^(.*)$/);
```

F. KWSRCH.PM

```
2c2
< # $Date: 2005/09/18 22:49:44 $
---
> # $Date: 2005/09/18 22:49:44 $
439d438
<
764c763,797
<                 print IDX "$results[$a]\n";
---
>
>                 my $block = "";
>                 my $meta  = "";
>                 my $fname = "";
>                 $block = $1 if $results[$a] =~ /^(\d+)\|/;
>
>                 if ($block ne "" && $ftype ne "raw" && $ftype ne "dls" ) {
>                     #figure out meta info and filename if they exists.
>                     local *OUT;
>                     Exec::exec_pipe(*OUT,
>                         "'$::TSKDIR/ifind' -f $ftype -d $block -o $offset -i
$imgtype $img");
>                     $meta = Exec::read_pipe_line(*OUT);
>                     close(OUT);
>
>                     if ($meta =~ /^($::REG_META)$/o) {
>                         $meta = $1;
>
>                         Exec::exec_pipe(*OUT,
>                             "'$::TSKDIR/ffind' -f $ftype -a -o $offset -i
$imgtype $img $meta");
>
>                         #just get the first line. only the first link to the
inode will be saved
>                         $fname = Exec::read_pipe_line(*OUT);
>                         chop $fname;
>
>                         close(OUT);
>                     }
>                     else {
>                         $meta = "";
>                     }
>                 }
>
>                 print IDX "$results[$a]";
>                 print IDX "|$meta" if ($meta ne "");
>                 print IDX "|$fname" if ($fname ne "");
>                 print IDX "\n";
904c937,957
<         unless (/^(\d+)\|(\d+)\|(.*)?$/) {
---
>         my $meta   = "";
>         my $fname  = "";
>         my $blk    = "";
>         my $off    = "";
>         my $str    = "";
>
>         if (/^(\d+)\|(\d+)\|(.*)\|(\d+)\|(.*)?$/) {
>             $blk    = $1;
```

```
>             $off    = $2;
>             $str    = $3;
>             $meta   = $4;
>             $fname  = $5;
>         }
>
>         elsif (/^(\d+)\|(\d+)\|(.*)?$/) {
>             $blk = $1;
>             $off = $2;
>             $str = $3;
>         }
>
>         else {
910,913d962
<         my $blk = $1;
<         my $off = $2;
<         my $str = $3;
<
929c978
<
---
>
930a980,1008
>
>                     #print meta info
>             print
>                 "$Fs::meta_str{$ftype}: "
> . "<a href=\"$::PROGNAME?mod=$::MOD_META&$Args::baseargs&"
> . "view=$Meta::STATS&meta=$meta\" target=\"content\">$meta</a><br>" if ($meta
ne "");
>
>             if ($fname ne "") {
>                 my $root = $Caseman::vol2mnt{$vol};
>
>                 #Print file info and make it red if it is deleted
>                 if ($fname =~ /^(\*)\s+\/*(.*)$/) {
>                     Print::print_output("File:
> <a href=\"$::PROGNAME?mod=$::MOD_FILE&view=$File::CONT_FR&$Args::baseargs
> &meta=$meta&dir=$fname&recmode=0\" target=\"content\">
> <font color=\"$::DEL_COLOR[0]\">$root$2</font></a> (deleted)<br>\n");
>                 }
>
>                 # If it starts with a '/' then it must be a file
>                 elsif ($fname =~ /^\/(.*)$/) {
>                     Print::print_output("File:
> <a href=\"$::PROGNAME?mod=$::MOD_FILE&view=$File::CONT_FR&$Args::baseargs
> &meta=$meta&dir=$fname&recmode=0\"
target=\"content\">$root$fname</a><br>\n");
>                 }
>                 # must be an error
>                 else {
>                     print "File: $fname<br>";
>                 }
>             }
```

G. MAIN.PM

```
2c2
< # $Date: 2005/09/18 22:49:44 $
---
> # $Date: 2005/09/18 22:49:44 $
62a63
> use Reports;
85c86
< $::REG_INVESTIG = '[\w]+';
---
> $::REG_INVESTIG = '[\w\.\-\s]+';
90c91
< $::REG_IMG_CONFIG    = '[\w\-\_\.\/ ]+';
---
> $::REG_IMG_CONFIG    = '[\w\-\_\.\/]+';
97a99
> $::REG_BMARK         = '[\w\s\-\_\.]+';
99c101
< $::REG_META = '[\d-]+';
---
> $::REG_META = '[\d\-]+';
124a127
> $::MOD_REPORTS = 12;
255a259,264
>     # The Case Reports module is handled seperatley because
>     # it may not have the host values
>     if ($module == $::MOD_REPORTS) {
>      return Reports::main();
>     }
>
351,353c360
<
<             # New modules can be added here
<
---
>             # addtional modules can go here
```

H. META.PM

```
2c2
< # $Date: 2005/09/18 22:49:44 $
---
> # $Date: 2005/09/18 22:49:44 $
205c205,221
<
---
>
>     #Determine names of files that point to this inode. Use for notes if one
link is found
>     my @pointedToBy;
>     local *OUT;
>     Exec::exec_pipe(*OUT,
>       "'$::TSKDIR/ffind' -f $ftype -a -o $offset -i $imgtype $img $meta");
>     while ($_ = Exec::read_pipe_line(*OUT)) {
>       chop;
>       if (/^(\*)\s+(\/*.*)$/) {
>         push @pointedToBy, "$2 (deleted)";
>       }
>       elsif (/^(.*)$/) {
>         push @pointedToBy, $1;
>       }
>     }
>     close(OUT);
>
244,245c260,263
<       . "&$Args::baseargs&meta=$meta&\" "
<       . "target=\"_blank\">"
---
>       . "&$Args::baseargs&meta=$meta";
>     print "&dir=$1"
>         if ( ($pointedToBy[0] =~ /^\/(.*)$/) && (scalar @pointedToBy == 1) );
>     print "&\" target=\"_blank\">"
255,260c273,278
<     if ($ftype =~ /fat/) {
<         print
< "<a href=\"$::PROGNAME?mod=$::MOD_META&view=$Meta::FFIND&$Args::baseargs&"
<           . "meta=$meta\" target=\"_blank\">Find File</a><br>";
<     }
<     else {
---
>     #if ($ftype =~ /fat/) {
>      #   print
> #"<a href=\"$::PROGNAME?mod=$::MOD_META&view=$Meta::FFIND&$Args::baseargs&"
>  #         . "meta=$meta\" target=\"_blank\">Find File</a><br>";
>   #  }
>    # else {
264,272c282,285
<         local *OUT;
<         Exec::exec_pipe(*OUT,
<             "'$::TSKDIR/ffind' -f $ftype -a -o $offset -i $imgtype $img
$meta");
<         my $cnt = 0;
<         while ($_ = Exec::read_pipe_line(*OUT)) {
<             chop;
<             if (/^(\*)\s+\/*(.*)$/) {
<                 Print::print_output(
< "<tt><font color=\"$::DEL_COLOR[0]\">$tmpr$2</font></tt> (deleted)<br>\n"
```

```
---
>         foreach my $f (@pointedToBy) {
>           if ($f =~ /^\/(.*\s+\(deleted\))$/) {
>             Print::print_output(
>                  "<tt><font
color=\"$::DEL_COLOR[0]\">$tmpr$1</font></tt><br>\n"
274,281c287,293
<             }
<             elsif (/^\/(.*)$/) {
<                 Print::print_output("<tt>$tmpr$1</tt><br>\n");
<             }
<             else {
<                 Print::print_output("$_<br>\n");
<             }
<             $cnt++;
---
>           }
>           elsif ($f =~ /^\/(.*)$/) {
>            Print::print_output("<tt>$tmpr$1</tt><br>\n");
>           }
>           else {
>            Print::print_output("$f<br>\n");
>           }
283,284c295,296
<         close(OUT);
<         if ($cnt == 0) {
---
>
>         if (scalar @pointedToBy == 0) {
288c300
<     }
---
>     #}
```

I. NOTES.PM

```
2c2
< # $Date: 2005/09/18 22:49:44 $
---
> # $Date: 2005/09/18 22:49:44 $
46a47,50
> $Notes::EDIT_NOTE     = 8;
> $Notes::EDIT_DOIT     = 9;
> $Notes::EDIT_NOTE_MAN = 10;
> $Notes::EDIT_DOIT_MAN = 11;
78a83,94
>     elsif ($view == $Notes::EDIT_NOTE) {
>         return edit_note();
>     }
>     elsif ($view == $Notes::EDIT_DOIT) {
>         return edit_doit();
>     }
>     elsif ($view == $Notes::EDIT_NOTE_MAN) {
>         return edit_note_man();
>     }
>     elsif ($view == $Notes::EDIT_DOIT_MAN) {
>         return edit_doit_man();
>     }
116c132
< "A note works like a bookmark and allows you to later find this data more
easily.<br><br>\n";
---
> "A Bookmark name allows you to categorize your notes. Choose an existing
bookmark name or create a new one.<br><br>\n";
119,120c135,152
<     print "<form action=\"$::PROGNAME\" method=\"get\">\n"
<       . "<textarea rows=10 cols=50 wrap=\"virtual\" name=\"note\">"
---
>     print "<form action=\"$::PROGNAME\" method=\"get\">\n";
>
>     my @bookmarks = read_bookmarks();
>     if (scalar @bookmarks > 0) {
>       print "<select name=\"bookmark\" size=\"1\">\n";
>
>       print "<option value=\"\" selected>Choose Bookmark</option>\n";
>
>       foreach my $i (@bookmarks) {
>      #make menu of bookmarks
>      print "<option value=\"$i\">$i</option>\n";
>       }
>       print "</select>  \n";
>     }
>     print "New: <input type=\"text\" name=\"bookmarknew\" size=\"15\"
maxlength=\"15\">"
>         . "<br><br\n";
>
>     print "<textarea rows=10 cols=50 wrap=\"virtual\" name=\"note\">"
179c211,456
<       . "  C-Time (<tt>$ctime</tt>)<br><hr><br>\n";
---
>       . "  C-Time (<tt>$ctime</tt>)<br><br><hr>\n";
>
>
>     # The OK Button
```

```
>     print "<br><input type=\"image\" src=\"pict/but_ok.jpg\" "
>       . "width=43 height=20 alt=\"Ok\" border=\"0\">\n</form>\n";
>
>     Print::print_html_footer();
>     return 0;
> }
>
> # window where user can edit a normal and sequencer note for a file
> # or meta data structure or block.
> sub edit_note {
>     Args::check_vol('vol');
>     Args::check_meta('meta') if (exists $Args::args{'meta'});
>     Args::check_block() if (exists $Args::args{'block'});
>     Args::check_len() if (exists $Args::args{'len'});
>
>
>     my $vol   = Args::get_vol('vol');
>     my $ftype = $Caseman::vol2ftype{$vol};
>     my $mnt   = $Caseman::vol2mnt{$vol};
>     my $meta  = Args::get_meta('meta') if (exists $Args::args{'meta'});
>     my $blk   = Args::get_block() if (exists $Args::args{'block'});
>     my $len   = Args::get_len() if (exists $Args::args{'len'});;
>     my $bmark_old = $Args::args{'bmark'};
>     my $note  = $Args::args{'note'};
>     chomp($note); chomp($note);
>     my $fname = "";
>     my $ntime = $Args::args{'ntime'};
>
>     # A file will have a 'dir' argument and a meta structure will not
>     if (exists $Args::args{'dir'}) {
>         $fname = "$mnt$Args::args{'dir'}";
>         Print::print_html_header("Edit Notes for file $fname");
>         print "<center><b>Edit a note for <tt>$fname</tt> ($meta):</b>"
>           . "<br><br>\n";
>     }
>     elsif (defined $meta) {
>         Print::print_html_header("Edit Notes for $Fs::meta_str{$ftype}
$meta");
>         print "<center><b>Edit a note for $Fs::meta_str{$ftype} $meta:</b>"
>           . "<br><br>\n";
>     }
>     elsif (defined $blk) {
>         Print::print_html_header("Edit Notes for $Fs::addr_unit{$ftype}
$blk");
>         print
>           "<center><b>Edit a note for $Fs::addr_unit{$ftype}
$blk</b><br><br>\n";
>     }
>     if ($ntime ne "") {
>         print "Note last edited on: $ntime<p><br></p>\n";
>     }
>     else {
>         print "No Note exists for this sequencer event<br></p>\n";
>     }
>     print
> "A Bookmark name allows you to categorize your notes. Choose an existing
bookmark name or create a new one.<br><br>\n";
>
>     # Setup the form
>     print "<form action=\"$::PROGNAME\" method=\"get\">\n";
>
>     my @bookmarks = read_bookmarks();
```

```
>     if (scalar @bookmarks > 0) {
>       print "<select name=\"bookmark\" size=\"1\">\n";
>
>       if ($bmark_old eq "") {
>           print "<option value=\"\" selected>Choose Bookmark</option>\n";
>       }
>       else {
>         print "<option value=\"\">Remove Bookmark</option>\n";
>       }
>       foreach my $i (@bookmarks) {
>      #make menu of bookmarks
>      print "<option value=\"$i\" ";
>         if ($i eq $bmark_old) {
>           print "selected";
>         }
>         print ">$i</option>\n";
>       }
>       print "</select>  \n";
>     }
>     print "New: <input type=\"text\" name=\"bookmarknew\" size=\"15\"
maxlength=\"15\">"
>         . "<br><br\n";
>
>     print "<textarea rows=10 cols=50 wrap=\"virtual\" name=\"note\">$note"
>       . "</textarea><br>\n"
>       . "<input type=\"hidden\" name=\"mod\" value=\"$::MOD_NOTES\">\n"
>       . "<input type=\"hidden\" name=\"view\" value=\"$Notes::EDIT_DOIT\">\n"
>       . "<input type=\"hidden\" name=\"vol\" value=\"$vol\">\n"
>       . "<input type=\"hidden\" name=\"bmark_old\" value=\"$bmark_old\">\n";
>     print "<input type=\"hidden\" name=\"meta\" value=\"$meta\">\n"
>       if (defined $meta);
>     print "<input type=\"hidden\" name=\"block\" value=\"$blk\">\n"
>       if (defined $blk);
>     print "<input type=\"hidden\" name=\"len\" value=\"$len\">\n"
>       if (defined $len);
>
>     print Args::make_hidden();
>
>     print "<input type=\"hidden\" name=\"ntime\" value=\"$ntime\">\n";
>     print "<input type=\"hidden\" name=\"dir\"
value=\"$Args::args{'dir'}\">\n"
>       if (exists $Args::args{'dir'});
>
>     # Option to add a normal note
>     print "<input type=\"checkbox\" name=\"norm_note\" value=\"1\"
CHECKED>\n"
>       . "  Add a Standard Note<br>\n";
>
>     if (!defined $blk) {
>       # Sequencer notes - which requires the MAC times for the files
>       if ("$Caseman::tz" ne "") {
>         $ENV{TZ} = $Caseman::tz;
>         POSIX: tzset();
>       }
>
>       my $img     = $Caseman::vol2path{$vol};
>       my $offset  = $Caseman::vol2start{$vol};
>       my $imgtype = $Caseman::vol2itype{$vol};
>
>       my $meta_int = $meta;
>       $meta_int = $1 if ($meta_int =~ /^(\d+)-\d+(-\d+)?$/);
>       local *OUT;
```

```
>       Exec::exec_pipe(*OUT,
>           "'$::TSKDIR/ils' -f $ftype -e  -o $offset -i $imgtype $img
$meta_int");
>
>       # Get the fourth line
>       my $tmp = Exec::read_pipe_line(*OUT);
>       $tmp = Exec::read_pipe_line(*OUT);
>       $tmp = Exec::read_pipe_line(*OUT);
>       $tmp = Exec::read_pipe_line(*OUT);
>       close(OUT);
>       unless ((defined $tmp)
>         && ($tmp =~ /^$::REG_META\|\w\|\d+\|\d+\|(\d+)\|(\d+)\|(\d+)\|/o))
>       {
>         Print::print_err("Error parsing 'ils' output<br>");
>       }
>
>       my $mtime = $1;
>       my $atime = $2;
>       my $ctime = $3;
>
>       my $mtime_nice = localtime($mtime);
>       my $atime_nice = localtime($atime);
>       my $ctime_nice = localtime($ctime);
>
>       # Print the Times
>       print "<br><hr><b>Add a Sequencer Event:</b><br><br>\n"
>         . "A sequencer event will be sorted based on the time so that event
reconstruction will be easier<br><br>\n";
>
>       # check if MAC times have previously been added as a seq event
>       my $checkm = 0;
>       my $checka = 0;
>       my $checkc = 0;
>
>       my $seq_notes_file = investig_seq_notes_fname();
>       if (-e $seq_notes_file) {
>         my ($msec, $mmin, $mhour, $mday, $mmon, $myear, $mwday, $myday,
$misdst) = localtime($mtime);
>         $myear += 1900;
>         $mmon  += 1;
>         $mday = "0$mday" if ($mday < 10);
>         $mhour = "0$mhour" if ($mhour < 10);
>         $mmin  = "0$mmin"  if ($mmin < 10);
>         $msec  = "0$msec"  if ($msec < 10);
>
>         my ($asec, $amin, $ahour, $aday, $amon, $ayear, $awday, $ayday,
$aisdst) = localtime($atime);
>         $ayear += 1900;
>         $amon  += 1;
>         $aday = "0$aday" if ($aday < 10);
>         $ahour = "0$ahour" if ($ahour < 10);
>         $amin  = "0$amin"  if ($amin < 10);
>         $asec  = "0$asec"  if ($asec < 10);
>
>         my ($csec, $cmin, $chour, $cday, $cmon, $cyear, $cwday, $cyday,
$cisdst) = localtime($ctime);
>         $cyear += 1900;
>         $cmon  += 1;
>         $cday = "0$cday" if ($cday < 10);
>         $chour = "0$chour" if ($chour < 10);
>         $cmin  = "0$cmin"  if ($cmin < 10);
>         $csec  = "0$csec"  if ($csec < 10);
```

```
>
>         open SEQ_NOTES, "$seq_notes_file" or die "Can't open log:
$seq_notes_file";
>         while (<SEQ_NOTES>) {
>             unless (
>
/^'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?($::REG_HOST)'?
,'?($::REG_VNAME)?'?,'?(.*?)'?,'?($::REG_META)?'?,'?(\d+)?'?,'?([\w\s]+)'?,'?(.
*?)',?'?(.*?)'?,?'?($::REG_BMARK)?'?\n/
>             )
>             {
>               Print::print_err("Error parsing sequence event entry: $_");
>             }
>             my $year = $1;
>             my $mon  = $2;
>             my $day  = $3;
>             my $hour = $4;
>             my $min  = $5;
>             my $sec  = $6;
>             my $host  = $7;
>          my $vol2  = "";
>             $vol2     = $8 if (defined $8);
>             my $fname2 = "";
>             $fname2    = $9 if (defined $9);
>             my $meta2  = "";
>             $meta2     = $10 if (defined $10);
>             my $note2  = $13;
>
>             #compare seq event entry w/ current note entry and see if they
match
>             if ($note2 =~ /^\[M-Time\](.*)$/)
>             {
>                 if (($myear eq $year) && ($mmon eq $mon) && ($mday eq $day)
>                     && ($mhour eq $hour) && ($mmin eq $min) && ($msec eq
$sec)
>                     && (Args::get_host() eq $host) && ($vol eq $vol2)
>                     && ($meta eq $meta2))
>                 {
>                     $checkm = 1;
>                 }
>             }
>             if ($note2 =~ /^\[A-Time\](.*)$/)
>             {
>                 if (($ayear eq $year) && ($amon eq $mon) && ($aday eq $day)
>                    && ($ahour eq $hour) && ($amin eq $min) && ($asec eq $sec)
>                    && (Args::get_host() eq $host) && ($vol eq $vol2)
>                    && ($meta eq $meta2))
>                 {
>                     $checka = 1;
>                 }
>             }
>             if ($note2 =~ /^\[C-Time\](.*)$/)
>             {
>                 if (($cyear eq $year) && ($cmon eq $mon) && ($cday eq $day)
>                    && ($chour eq $hour) && ($cmin eq $min) && ($csec eq $sec)
>                    && (Args::get_host() eq $host) && ($vol eq $vol2)
>                    && ($meta eq $meta2))
>                 {
>                     $checkc = 1;
>                 }
>             }
>         }
```

```
>       }
>       print "<input type=\"checkbox\" name=\"mtime\" value=\"1\" ";
>       print "CHECKED" if $checkm;
>       print ">  M-Time (<tt>$mtime_nice</tt>)<br>\n";
>
>       print "<input type=\"checkbox\" name=\"atime\" value=\"1\" ";
>       print "CHECKED" if ($checka);
>       print ">  A-Time (<tt>$atime_nice</tt>)<br>\n"
>         . "<input type=\"checkbox\" name=\"ctime\" value=\"1\" ";
>       print "CHECKED" if ($checkc);
>       print ">  C-Time (<tt>$ctime_nice</tt>)<br><br><hr>\n";
>     }
188a466,1121
> # Edit a manually added sequencer event
> sub edit_note_man {
>     my $bmark_old = $Args::args{'bmark'};
>     my $note  = $Args::args{'note'};
>     chomp($note); chomp($note);
>
>     Print::print_html_header("Edit a Manually Entered Event");
>     print "<center><b>Edit a Manually Entered Event<br><br>\n";
>
>     # Setup the form
>     print "<form action=\"$::PROGNAME\" method=\"get\">\n";
>
>     my @bookmarks = read_bookmarks();
>     if (scalar @bookmarks > 0) {
>       print "<select name=\"bookmark\" size=\"1\">\n";
>
>       if ($bmark_old eq "") {
>           print "<option value=\"\" selected>Choose Bookmark</option>\n";
>       }
>       else {
>         print "<option value=\"\">Remove Bookmark</option>\n";
>       }
>       foreach my $i (@bookmarks) {
>      #make menu of bookmarks
>      print "<option value=\"$i\" ";
>         if ($i eq $bmark_old) {
>           print "selected";
>         }
>         print ">$i</option>\n";
>       }
>       print "</select>  \n";
>     }
>     print "New: <input type=\"text\" name=\"bookmarknew\" size=\"15\"
maxlength=\"15\">"
>         . "<br><br\n";
>
>     print "<textarea rows=10 cols=50 wrap=\"virtual\" name=\"note\">$note"
>       . "</textarea><br>\n"
>       . "<input type=\"hidden\" name=\"mod\" value=\"$::MOD_NOTES\">\n"
>       . "<input type=\"hidden\" name=\"view\"
value=\"$Notes::EDIT_DOIT_MAN\">\n"
>       . "<input type=\"hidden\" name=\"bmark_old\" value=\"$bmark_old\">\n";
>     print Args::make_hidden();
>
>     my ($year, $mon, $day, $hour, $min, $sec, $host, $source, $note2);
>     my $seq_notes_file = investig_seq_notes_fname();
>     if (-e $seq_notes_file) {
>
```

```
>       open SEQ_NOTES, "$seq_notes_file" or die "Can't open log:
$seq_notes_file";
>       while (<SEQ_NOTES>) {
>           unless (
>
/^'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?($::REG_HOST)'?
,'?($::REG_VNAME)?'?,'?(.*?)'?,'?($::REG_META)?'?,'?(\d+)?'?,'?([\w\s]+)'?,'?(.
*?)',?'?(.*?)'?,?'?($::REG_BMARK)?'?\n/
>           )
>           {
>             Print::print_err("Error parsing sequence event entry: $_");
>           }
>           $year = $1;
>           $mon  = $2;
>           $day  = $3;
>           $hour = $4;
>           $min  = $5;
>           $sec  = $6;
>           $host  = $7;
>           $source = "";
>           $source    = $12 if (defined $12);
>           $note2  = $13;
>
>       }
>     }
>     print "<input type=\"hidden\" name=\"year_old\" value=\"$year\">\n";
>     print "<input type=\"hidden\" name=\"mon_old\" value=\"$mon\">\n";
>     print "<input type=\"hidden\" name=\"day_old\" value=\"$day\">\n";
>     print "<input type=\"hidden\" name=\"hour_old\" value=\"$hour\">\n";
>     print "<input type=\"hidden\" name=\"min_old\" value=\"$min\">\n";
>     print "<input type=\"hidden\" name=\"sec_old\" value=\"$sec\">\n";
>     print "<input type=\"hidden\" name=\"source_old\" value=\"$source\">\n";
>     print "<input type=\"hidden\" name=\"note_old\" value=\"$note2\">\n";
>
>    # Month
>     print "<p>Date: " . "<select name=\"mon\" size=\"1\">\n";
>     for my $i (1 .. 12) {
>         if ($i == $mon) {
>             print "<option value=$i selected>$::d2m[$i]</option>\n";
>         }
>         else {
>             print "<option value=\"$i\">$::d2m[$i]</option>\n";
>         }
>     }
>
>     print "</select>\n";
>
>     # Day
>     print " <select name=\"day\" size=\"1\">\n";
>     for my $i (1 .. 31) {
>         my $dstr = $i;
>         $dstr = "0$i" if ($i < 10);
>
>         if ($day eq $dstr) {
>             print "<option value=\"$dstr\" selected>$dstr</option>\n";
>         }
>         else {
>             print "<option value=\"$dstr\">$dstr</option>\n";
>         }
>     }
>     print "</select>\n";
>
```

```
>     # Year
>     print " <input type=\"text\" value=\"$year\" "
>       . "name=\"year\" size=\"6\">\n";
>
>     # Hour
>     print "  <select name=\"hour\" size=\"1\">\n";
>     for my $i (0 .. 23) {
>         my $hstr = $i;
>         $hstr = "0$i" if ($i < 10);
>
>         if ($hour eq $hstr) {
>             print "<option value=\"$hstr\" selected>$hstr</option>\n";
>         }
>         else {
>             print "<option value=\"$hstr\">$hstr</option>\n";
>         }
>     }
>     print "</select>\n";
>
>     # Min
>     print ":<select name=\"min\" size=\"1\">\n";
>     for my $i (0 .. 59) {
>         my $mstr = $i;
>         $mstr = "0$i" if ($i < 10);
>         if ($min eq $mstr) {
>             print "<option value=\"$mstr\" selected>$mstr</option>\n";
>         }
>         else {
>             print "<option value=\"$mstr\">$mstr</option>\n";
>         }
>     }
>     print "</select>\n";
>
>     # Sec
>     print ":<select name=\"sec\" size=\"1\">\n";
>     for my $i (0 .. 59) {
>         my $sstr = $i;
>         $sstr = "0$i" if ($i < 10);
>
>         if ($sec eq $sstr) {
>             print "<option value=\"$sstr\" selected>$sstr</option>\n";
>         }
>         else {
>             print "<option value=\"$sstr\">$sstr</option>\n";
>         }
>     }
>     print "</select>\n";
>
>     # Type
>     print "<br><br>Source of Event: <select name=\"src\" size=1>\n"
>       . "<option value=\"firewall\" "
>       . ( ($source eq "firewall") ? "selected" : "")
>       . ">firewall</option>\n"
>       . "<option value=\"ids\" "
>       . ( ($source eq "ids") ? "selected" : "")
>       . ">ids</option>\n"
>       . "<option value=\"isp\" "
>       . ( ($source eq "isp") ? "selected" : "")
>       . ">isp</option>\n"
>       . "<option value=\"log\" "
>       . ( ($source eq "log") ? "selected" : "")
>       . ">log</option>\n"
```

```
>       . "<option value=\"other\" "
>       . ( ($source eq "other") ? "selected" : "")
>       . ">other</option>\n"
>       . "<option value=\"person\" "
>       . ( ($source eq "person") ? "selected" : "")
>       . ">person</option>\n"
>       . "</select>\n";
>
>     # The OK Button
>     print "<br><br><input type=\"image\" src=\"pict/but_ok.jpg\" "
>       . "width=43 height=20 alt=\"Ok\" border=\"0\">\n</form>\n";
>
>     Print::print_html_footer();
>     return 0;
> }
>
> # update notes or sequencer events file after edit
> sub edit_doit {
>     Args::check_vol('vol');
>     Args::check_meta('meta') if (exists $Args::args{'meta'});
>     Args::check_block() if (exists $Args::args{'block'});
>     Args::check_len() if (exists $Args::args{'len'});
>     Args::check_note();
>     Args::check_bmark();
>
>     # Get rid of carriage returns that Netscape adds
>     $Args::args{'note'} =~ tr/\r//d;
>
>     my $vol = Args::get_vol('vol');
>     my $mnt = $Caseman::vol2mnt{$vol};
>     my $bmark = Args::get_bmark();
>     my $bmark_old = $Args::args{'bmark_old'};
>
>     my $ftype  = $Caseman::vol2ftype{$vol};
>     my $img_sh = $Caseman::vol2sname{$vol};
>     my $meta   = Args::get_meta('meta') if (exists $Args::args{'meta'});
>     my $blk    = Args::get_block() if (exists $Args::args{'block'});
>     my $len    = Args::get_len() if (exists $Args::args{'len'});
>
>     my $img     = $Caseman::vol2path{$vol};
>     my $offset  = $Caseman::vol2start{$vol};
>     my $imgtype = $Caseman::vol2itype{$vol};
>
>     my $ntime = $Args::args{'ntime'};
>     my $fname = "";
>     my $type  = "";
>
>     if (exists $Args::args{'dir'}) {
>       $Args::args{'dir'} .= "/"
>       if ($Args::args{'dir'} eq "");
>         $fname = "$mnt$Args::args{'dir'}";
>
>         if (($Args::args{'dir'} =~ /\/$/) || ($Args::args{'dir'} eq "")) {
>             Print::log_host_inv(
>                 "$img_sh: Editing note for directory $fname ($meta), Bookmark
$bmark");
>             $type = "dir";
>         }
>         else {
>             Print::log_host_inv(
>                 "$img_sh: Editing note for file $fname ($meta), Bookmark
$bmark");
```

```
>             $type = "file";
>         }
>     }
>
>     elsif (exists $Args::args{'meta'}) {
>         Print::log_host_inv(
>             "$img_sh: Editing note for $Fs::meta_str{$ftype} $meta, Bookmark
$bmark");
>         $type = "$Fs::meta_str{$ftype}";
>     }
>
>     elsif (exists $Args::args{'block'}) {
>         Print::log_host_inv(
>         "$img_sh: Editing note for $Fs::addr_unit{$ftype} $blk, Bookmark
$bmark");
>     }
>
>     my $read  = investig_notes_fname();
>     my $write = $read . "-" . rand();
>
>     if (-e $read) {
>       unless (open(READ, "<$read")) {
>          Print::print_check_err("Error opening $read");
>       }
>     }
>     unless (open(WRITE, ">$write")) {
>         Print::print_check_err("Error opening $write");
>     }
>
>     #write contents of notes file to new file, skipping the note to be edited
>     #detecting a note for removal based on note creation time
>     if (-e $read) {
>       my $cpy = 1;
>       while (<READ>) {
>           if (/^($ntime)$/ && $ntime ne "") {
>               $cpy = 0;
>           }
>           elsif ($cpy eq 0 && /^\-+$/) {
>               $cpy = 1;
>           }
>           elsif ($cpy eq 1) {
>               print WRITE "$_";
>           }
>       }
>     }
>
>     Print::print_html_header("Writing a note / event");
>     my $new_ntime = localtime();
>     my $mtime = "";
>     my $atime = "";
>     my $ctime = "";
>
>     if (defined $meta) {
>       # Get the times for the meta
>       # Set the timezone to the host zone
>       if ("$Caseman::tz" ne "") {
>         $ENV{TZ} = "$Caseman::tz";
>         POSIX::tzset();
>       }
>
>       my $meta_int = $meta;
>       $meta_int = $1 if ($meta_int =~ /^(\d+)-\d+(-\d+)?$/);
```

```
>       local *OUT;
>       Exec::exec_pipe(*OUT,
>         "'$::TSKDIR/ils' -f $ftype -e -o $offset -i $imgtype $img
$meta_int");
>
>       # Skip to the fourth line
>       my $tmp = Exec::read_pipe_line(*OUT);
>       $tmp = Exec::read_pipe_line(*OUT);
>       $tmp = Exec::read_pipe_line(*OUT);
>       $tmp = Exec::read_pipe_line(*OUT);
>       unless ((defined $tmp)
>         && ($tmp =~ /^$::REG_META\|\w\|\d+\|\d+\|(\d+)\|(\d+)\|(\d+)\|/o))
>       {
>         Print::print_err("Error parsing 'ils' output<br>");
>       }
>       $mtime = $1;
>       $atime = $2;
>       $ctime = $3;
>       close(OUT);
>     }
>
>     # now add the edited note back in
>     if ((exists $Args::args{'norm_note'}) && ($Args::args{'norm_note'} == 1))
{
>         print "Note updated in $read<p>\n\n";
>
>         # Date
>         print WRITE "$new_ntime\n";
>         print "$ntime<br>\n";
>
>      # Bookmark
>      if ($bmark ne "") {
>        print WRITE "Bookmark: $bmark\n";
>        print "Bookmark: $bmark<br>\n";
>      }
>
>         # We have a file name
>         if ($fname ne "") {
>             if ($type eq 'dir') {
>                 print WRITE "Directory: $fname\n";
>                 print "Directory: $fname<br>\n";
>             }
>             else {
>                 print WRITE "File: $fname\n";
>                 print "File: $fname<br>\n";
>             }
>         }
>         if (defined $meta && $meta ne "") {
>             print WRITE "Volume: $vol  Meta: $meta\n";
>             print "Volume: $vol  Meta: $meta<br>\n";
>         }
>
>         if (defined $blk && $blk ne "" && defined $len && $len ne "") {
>             print WRITE "Volume: $vol  $Fs::addr_unit{$ftype}: $blk  Len:
$len\n";
>             print "Volume: $vol  $Fs::addr_unit{$ftype}: $blk  Len:
$len<br>\n";
>         }
>
>         if (defined $meta) {
>           print WRITE "M-time: " . localtime($mtime) . "\n";
>           print "M-time: " . localtime($mtime) . "<br>\n";
```

```
>           print WRITE "A-time: " . localtime($atime) . "\n";
>           print "A-time: " . localtime($atime) . "<br>\n";
>           print WRITE "C-time: " . localtime($ctime) . "\n";
>           print "C-time: " . localtime($ctime) . "<br>\n";
>         }
>
>         # The actual notes and a line at the bottom
>         print WRITE "\n$Args::args{'note'}\n\n" . "-" x 62 . "\n";
>         print "<p>$Args::args{'note'}<p>";
>     }
>
>     close(READ);
>     close(WRITE);
>
>     unless (rename $write, $read) {
>         print STDERR "Error renaming temp notes file\n";
>     }
>
>     #Edit sequencer events
>     $read = investig_seq_notes_fname();
>     $write = $read . "-" . rand();
>     if (defined $meta && -e $read) {
>       unless (open(READ, "<$read")) {
>         Print::print_check_err("Error opening $read");
>       }
>       unless (open(WRITE, ">$write")) {
>         Print::print_check_err("Error opening $write");
>       }
>
>       # m-time
>       my ($msec, $mmin, $mhour, $mday, $mmon, $myear, $mwday, $myday,
$misdst) = localtime($mtime);
>       $myear += 1900;
>       $mmon  += 1;
>       $mday = "0$mday" if ($mday < 10);
>       $mhour = "0$mhour" if ($mhour < 10);
>       $mmin  = "0$mmin"  if ($mmin < 10);
>       $msec  = "0$msec"  if ($msec < 10);
>
>       # a-time
>       my ($asec, $amin, $ahour, $aday, $amon, $ayear, $awday, $ayday,
$aisdst) = localtime($atime);
>       $ayear += 1900;
>       $amon  += 1;
>       $aday = "0$aday" if ($aday < 10);
>       $ahour = "0$ahour" if ($ahour < 10);
>       $amin  = "0$amin"  if ($amin < 10);
>       $asec  = "0$asec"  if ($asec < 10);
>
>       # c-time
>       my ($csec, $cmin, $chour, $cday, $cmon, $cyear, $cwday, $cyday,
$cisdst) = localtime($ctime);
>       $cyear += 1900;
>       $cmon  += 1;
>       $cday = "0$cday" if ($cday < 10);
>       $chour = "0$chour" if ($chour < 10);
>       $cmin  = "0$cmin"  if ($cmin < 10);
>       $csec  = "0$csec"  if ($csec < 10);
>
>       while (<READ>) {
>         unless (
```

```
>
/^'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?($::REG_HOST)'?
,'?($::REG_VNAME)?'?,'?(.*?)'?,'?($::REG_META)?'?,'?(\d+)?'?,'?([\w\s]+)'?,'?(.
*?)',?'?(.*?)'?,?'?($::REG_BMARK)?'?\n/
>         )
>         {
>           Print::print_err("Error parsing sequence event entry: $_");
>         }
>         my $year = $1;
>         my $mon  = $2;
>         my $day  = $3;
>         my $hour = $4;
>         my $min  = $5;
>         my $sec  = $6;
>         my $host  = $7;
>      my $vol2  = "";
>         $vol2     = $8 if (defined $8);
>         my $fname2 = "";
>         $fname2    = $9 if (defined $9);
>         my $meta2  = "";
>         $meta2     = $10 if (defined $10);
>         my $note2  = $13;
>         my $bmark2 = "";
>      $bmark2    = $15 if (defined $15);
>
>         #compare seq event entry w/ current note entry and see if they match
>         if ($note2 =~ /^\[M-Time\](.*)$/)
>         {
>            unless (($myear eq $year) && ($mmon eq $mon) && ($mday eq $day)
>                 && ($mhour eq $hour) && ($mmin eq $min) && ($msec eq $sec)
>                 && (Args::get_host() eq $host) && ($vol eq $vol2)
>                 && ($meta eq $meta2) && ($bmark_old eq $bmark2))
>            {
>               print WRITE "$_";
>            }
>         }
>         elsif ($note2 =~ /^\[A-Time\](.*)$/)
>         {
>            unless (($ayear eq $year) && ($amon eq $mon) && ($aday eq $day)
>                 && ($ahour eq $hour) && ($amin eq $min) && ($asec eq $sec)
>                 && (Args::get_host() eq $host) && ($vol eq $vol2)
>                 && ($meta eq $meta2) && ($bmark_old eq $bmark2))
>            {
>               print WRITE "$_";
>            }
>         }
>         elsif ($note2 =~ /^\[C-Time\](.*)$/)
>         {
>            unless (($cyear eq $year) && ($cmon eq $mon) && ($cday eq $day)
>                 && ($chour eq $hour) && ($cmin eq $min) && ($csec eq $sec)
>                 && (Args::get_host() eq $host) && ($vol eq $vol2)
>                 && ($meta eq $meta2) && ($bmark_old eq $bmark2))
>            {
>               print WRITE "$_";
>            }
>         }
>         else {
>             print WRITE "$_";
>         }
>       }
>
>       # Create a sequencer event - if there are any
```

```
>       if (((exists $Args::args{'mtime'}) && ($Args::args{'mtime'} == 1))
>           || ((exists $Args::args{'atime'}) && ($Args::args{'atime'} == 1))
>           || ((exists $Args::args{'ctime'}) && ($Args::args{'ctime'} == 1)))
>       {
>         # Get rid of the carriage returns
>         $Args::args{'note'} =~ s/\n/<br>/gs;
>         # Replace single quote chars w/ html escape sequence. Makes parsing
seq.notes file easier
>         $Args::args{'note'} =~ s/'/&#39/gs;
>
>         #determine file type
>         Exec::exec_pipe(*OUT,
>             "'$::TSKDIR/icat' -f $ftype -r  -o $offset -i $imgtype $img $meta
| '$::FILE_EXE' -z -b -"
>         );
>         my $apptype = Exec::read_pipe_line(*OUT);
>         close(OUT);
>
>         $apptype = "" if ((!defined $apptype));
>         # Replace single quote chars w/ html escape sequence. Makes parsing
seq.notes file easier
>         $apptype =~ s/'/&#39/gs;
>         chomp $apptype; #get rid of newline
>
>         # m-time
>         if ((exists $Args::args{'mtime'}) && ($Args::args{'mtime'} == 1)) {
>           print WRITE "'$myear','$mmon','$mday','$mhour','$mmin','$msec',"
>             . "'$Args::args{'host'}','$vol','$fname','$meta','',"
>             . "'$type','[M-Time]$Args::args{'note'}','$apptype','$bmark'\n";
>
>           Print::log_host_inv(
>             "$img_sh: M-Time note updated for meta $Args::args{'meta'},
Bookmark $bmark");
>           print "M-Time sequence event updated<br>\n";
>         }
>
>         if ((exists $Args::args{'atime'}) && ($Args::args{'atime'} == 1)) {
>           print WRITE "'$ayear','$amon','$aday','$ahour','$amin','$asec',"
>             . "'$Args::args{'host'}','$vol','$fname','$meta','',"
>             . "'$type','[A-Time]$Args::args{'note'}','$apptype','$bmark'\n";
>
>           Print::log_host_inv(
>             "$img_sh: A-Time note updated for meta $Args::args{'meta'},
Bookmark $bmark");
>           print "A-Time sequence event updated<br>\n";
>         }
>
>         if ((exists $Args::args{'ctime'}) && ($Args::args{'ctime'} == 1)) {
>           print WRITE "'$cyear','$cmon','$cday','$chour','$cmin','$csec',"
>             . "'$Args::args{'host'}','$vol','$fname','$meta','',"
>             . "'$type','[C-Time]$Args::args{'note'}','$apptype','$bmark'\n";
>
>           Print::log_host_inv(
>              "$img_sh: C-Time note updated for meta $Args::args{'meta'},
Bookmark $bmark");
>           print "C-Time sequence event updated<br>\n";
>         }
>       }
>       close(READ);
>       close(WRITE);
>
>       unless (rename $write, $read) {
```

```
>         print STDERR "Error renaming temp notes file\n";
>       }
>     }
>
>     print "<p><strong>Done updating note / sequencer event.<br>"
>         . "Close this window and refresh the Notes or Event Sequencer
page</strong></p>\n";
>     Print::print_html_footer();
>     return 0;
> }
>
> # update sequencer events file after edit if sequencer event was added
manually
> sub edit_doit_man {
>     Args::check_note();
>     Args::check_bmark();
>
>     # Get rid of carriage returns that Netscape adds
>     $Args::args{'note'} =~ tr/\r//d;
>
>     my $bmark = Args::get_bmark();
>     my $bmark_old = $Args::args{'bmark_old'};
>
>     #Edit sequencer events
>     $read = investig_seq_notes_fname();
>     $write = $read . "-" . rand();
>     if (-e $read) {
>       unless (open(READ, "<$read")) {
>         Print::print_check_err("Error opening $read");
>       }
>       unless (open(WRITE, ">$write")) {
>         Print::print_check_err("Error opening $write");
>       }
>
>       while (<READ>) {
>         unless (
>
/^'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?($::REG_HOST)'?
,'?($::REG_VNAME)?'?,'?(.*?)'?,'?($::REG_META)?'?,'?(\d+)?'?,'?([\w\s]+)'?,'?(.
*?)',?'?(.*?)'?,?'?($::REG_BMARK)?'?\n/
>         )
>         {
>           Print::print_err("Error parsing sequence event entry: $_");
>         }
>         my $year = $1;
>         my $mon  = $2;
>         my $day  = $3;
>         my $hour = $4;
>         my $min  = $5;
>         my $sec  = $6;
>         my $source = $12;
>         my $note  = $13;
>         my $bmark2 = "";
>      $bmark2    = $15 if (defined $15);
>
>         #compare seq event entry w/ current note entry and see if they match
>         unless (($Args::args{'year_old'} eq $year) && ($Args::args{'mon_old'}
eq $mon)
>             && ($Args::args{'day_old'} eq $day) && ($Args::args{'hour_old'}
eq $hour)
>             && ($Args::args{'min_old'} eq $min) && ($Args::args{'sec_old'} eq
$sec)
```

```
>             && ($Args::args{'source_old'} eq $source) && ($bmark_old eq
$bmark2))
>         {
>             print WRITE "$_";
>         }
>       }
>
>       Print::print_html_header("Writing Sequencer Event");
>
>       # Get rid of carriage returns that Netscape adds
>       $Args::args{'note'} =~ tr/\r//d;
>       $Args::args{'note'} =~ s/\n/<br>/gs;
>       # Replace single quote chars w/ html escape sequence. Makes parsing
seq.notes file easier
>       $Args::args{'note'} =~ s/'/&#39/gs;
>
>       if ($Args::args{'note'} eq "") {
>         Print::print_err("A comment must be given for the event<br>\n"
>               . "<p><a
href=\"$::PROGNAME?mod=$::MOD_NOTES&view=$Notes::READ_SEQ&$Args::baseargs\">"
>               . "<img border=0 src=\"pict/menu_b_ok.jpg\" "
>               . "width=167 height=20></a>");
>       }
>
>       # Check the args and add them to the final string that will be written
>       my $str = "";
>       unless ((exists $Args::args{'year'})
>         && ($Args::args{'year'} =~ /^(\d\d\d\d)$/))
>       {
>         Print::print_err("Invalid year<br>");
>       }
>       $str .= "'$1',";
>
>       unless ((exists $Args::args{'mon'}) && ($Args::args{'mon'} =~
/^(\d\d?)$/))
>       {
>         Print::print_err("Invalid month<br>");
>       }
>       $str .= "'$1',";
>
>       unless ((exists $Args::args{'day'}) && ($Args::args{'day'} =~
/^(\d\d?)$/))
>       {
>         Print::print_err("Invalid day<br>");
>       }
>       $str .= "'$1',";
>
>       unless ((exists $Args::args{'hour'})
>         && ($Args::args{'hour'} =~ /^(\d\d?)$/))
>       {
>         Print::print_err("Invalid hour<br>");
>       }
>       $str .= "'$1',";
>
>       unless ((exists $Args::args{'min'}) && ($Args::args{'min'} =~
/^(\d\d?)$/))
>       {
>         Print::print_err("Invalid min<br>");
>       }
>       $str .= "'$1',";
>
```

```
>       unless ((exists $Args::args{'sec'}) && ($Args::args{'sec'} =~
/^(\d\d?)$/))
>       {
>         Print::print_err("Invalid sec<br>");
>       }
>       $str .= "'$1',";
>
>       # There are no image, meta, file name, or data unit for this type
>       $str .= "'$Args::args{'host'}','','','','',";
>
>       unless ((exists $Args::args{'src'}) && ($Args::args{'src'} =~
/^(\w+)$/)) {
>         Print::print_err("Invalid source<br>");
>       }
>       $str .= "'$1','$Args::args{'note'}','','$bmark'\n";
>
>       print WRITE $str;
>
>       close(READ);
>       close(WRITE);
>
>       unless (rename $write, $read) {
>         print STDERR "Error renaming temp notes file\n";
>       }
>     }
>
>     print "<p><strong>Done updating sequencer event.<br>"
>         . "Close this window and refresh the Event Sequencer
page</strong></p>\n";
>     Print::print_html_footer();
>     return 0;
> }
>
202,203c1135,1149
<       . "A note works like a bookmark and allows you to later find this data
more easily.<br><br>\n"
<       . "<form action=\"$::PROGNAME\" method=\"get\">\n"
---
>     . "<form action=\"$::PROGNAME\" method=\"get\">\n"
>     . "A Bookmark name allows you to categorize your notes. Choose an
existing bookmark name or create a new one.<br><br>\n";
>
>     my @bookmarks = read_bookmarks();
>     if (scalar @bookmarks > 0) {
>       print "<select name=\"bookmark\" size=\"1\">\n"
>           . "<option value=\"\" selected>Choose Bookmark</option>\n";
>       foreach my $i (@bookmarks) {
>      #make menu of bookmarks
>      print "<option value=\"$i\">$i</option>\n";
>       }
>       print "</select>  \n";
>     }
>     print "New: <input type=\"text\" name=\"bookmarknew\" size=\"15\"
maxlength=\"15\">"
>       . "<br><br\n"
226a1173
>     Args::check_bmark();
234a1182
>     my $bmark  = Args::get_bmark();
237c1185
<         "$img_sh: Creating note for $Fs::addr_unit{$ftype} $block");
---
```

```
>         "$img_sh: Creating note for $Fs::addr_unit{$ftype} $block, Bookmark
$bmark");
247c1195,1203
<
---
>
>     # Bookmark
>     print NOTES "Bookmark: ";
>     print "Bookmark: ";
>     if ($bmark ne "") {
>       print NOTES "$bmark\n";
>       print "$bmark<br>\n";
>     }
>
271a1228
>     Args::check_bmark();
277a1235
>     my $bmark = Args::get_bmark();
297c1255
<                 "$img_sh: Creating note for directory $fname ($meta)");
---
>                 "$img_sh: Creating note for directory $fname ($meta),
Bookmark $bmark");
302c1260
<                 "$img_sh: Creating note for file $fname ($meta)");
---
>                 "$img_sh: Creating note for file $fname ($meta), Bookmark
$bmark");
309c1267
<             "$img_sh: Creating note for $Fs::meta_str{$ftype} $meta");
---
>             "$img_sh: Creating note for $Fs::meta_str{$ftype} $meta, Bookmark
$bmark");
314a1273,1274
>     my $ntime = localtime();  #Moved this here so correct time of note is
reported
>
351,353c1311,1320
<         my $tmp = localtime();
<         print NOTES "$tmp\n";
<         print "$tmp\n";
---
>         print NOTES "$ntime\n";
>         print "$ntime\n";
>
>         # Bookmark
>         print NOTES "Bookmark: ";
>         print "Bookmark: ";
>      if ($bmark ne "") {
>        print NOTES "$bmark\n";
>        print "$bmark<br>\n";
>      }
403c1370,1386
<
---
>     # Replace single quote chars w/ html escape sequence. Makes parsing
seq.notes file easier
>     $Args::args{'note'} =~ s/'/&#39/gs;
>
>     #determine file type
>     Exec::exec_pipe(*OUT,
```

```
> "'$::TSKDIR/icat' -f $ftype -r  -o $offset -i $imgtype $img $meta |
'$::FILE_EXE' -z -b -"
>         );
>     my $apptype = Exec::read_pipe_line(*OUT);
>     close(OUT);
>
>     $apptype = ""
>       if ((!defined $apptype));
>
>     # Replace single quote chars w/ html escape sequence. Makes parsing
seq.notes file easier
>     $apptype =~ s/'/&#39/gs;
>     chomp $apptype; #get rid of newline
>
420c1403
<           . "'$type','[M-Time]$Args::args{'note'}'\n";
---
>           . "'$type','[M-Time]$Args::args{'note'}','$apptype','$bmark'\n";
423c1406
<             "$img_sh: M-Time note added for meta $Args::args{'meta'}");
---
>             "$img_sh: M-Time note added for meta $Args::args{'meta'},
Bookmark $bmark");
439c1422
<           . "'$type','[A-Time]$Args::args{'note'}'\n";
---
>           . "'$type','[A-Time]$Args::args{'note'}','$apptype','$bmark'\n";
442c1425
<             "$img_sh: A-Time note added for meta $Args::args{'meta'}");
---
>             "$img_sh: A-Time note added for meta $Args::args{'meta'},
Bookmark $bmark");
458c1441
<           . "'$type','[C-Time]$Args::args{'note'}'\n";
---
>           . "'$type','[C-Time]$Args::args{'note'}','$apptype','$bmark'\n";
461c1444
<             "$img_sh: C-Time note added for meta $Args::args{'meta'}");
---
>             "$img_sh: C-Time note added for meta $Args::args{'meta'},
Bookmark $bmark");
501a1485,1488
>     my $vol = "";
>     my $blk = "";
>     my $len = "";
>     my $meta = "";
503a1491,1494
>     my $fname = "";
>     my $bmark = "";
>     my $ntime = "";
>     my $note = "";
513,516d1503
<         # we need to extract mnt from here
<         $file = $1 if (/^File: (.*)$/);
<         $dir  = $1 if (/^Directory: (.*)$/);
<
518a1506,1528
>           #Edit button
>
>           print "<a href=\"$::PROGNAME?$Args::baseargs_novol"
>               . "&mod=$::MOD_NOTES&view=$Notes::EDIT_NOTE"
>               . "&vol=$vol";
```

```
>           if ($meta ne "") {
>             print "&meta=$meta";
>             if ($fname ne "") {
>               print "&dir=$fname";
>             }
>           }
>           elsif ($blk ne "") {
>             print "&block=$blk";
>             if ($len ne "") {
>                 print "&len=$len";
>             }
>           }
>           print "&bmark=$bmark&note="
>               . Args::url_encode($note) . "&ntime=$ntime\"
target=\"_blank\">"
>               . "<img src=\"pict/but_edit.jpg\" alt=\"edit note\" "
>               . "width=45 height=22 border=\"0\"></a><br><hr>";
>
>             $meta = "";
520a1531,1536
>             $bmark = "";
>             $note = "";
>             $fname = "";
>             $blk = "";
>             $len = "";
>             $ntime = "";
529,531d1544
<         else {
<             print "$_<br>";
<         }
533c1546,1560
<         if (/^Volume: ($::REG_VNAME)   Meta: ([0-9\-]+)/o) {
---
>         elsif (/^Bookmark:\s+(.*)$/) {
>           $bmark = $1;
>           print "$_<br>";
>         }
>         # we need to extract mnt from here
>         elsif (/^File: (.*)$/) {
>           $file = $1;
>           print "$_<br>";
>         }
>         elsif (/^Directory: (.*)$/) {
>           $dir  = $1;
>           print "$_<br>";
>         }
>
>         elsif (/^Volume:\s+($::REG_VNAME)\s+Meta:\s+([0-9\-]+)/o) {
535a1563
>             print "$_<br>";
540d1567
<
543c1570
<                 my $fname = "";
---
>                 $fname = "";
547c1574
<                   . "&vol=$vol&meta=$meta&dir=$fname\" target=\"_blank\">"
---
>                   . "&vol=$vol&meta=$meta&dir=$fname&bmark=\"
target=\"_blank\">"
579c1606
```

```
<         elsif (/^Volume: ($::REG_VNAME)   \w+: ([0-9]+)  Len: (\d+)/o) {
---
>         elsif (/^Volume:\s+($::REG_VNAME)\s+\w+:\s+([0-9]+)\s+Len: (\d+)/o) {
582a1610
>             print "$_<br>";
591a1620,1632
>         #time note created
>         elsif (/^\w+\s+\w+\s+\d{1,2}\s+\d\d:\d\d:\d\d\s+\d{4}$/) {
>             $ntime = "$_";
>             print "$_<br>";
>         }
>         #save note content; ignore MAC times
>         elsif (! /^[MAC]-time:.*$/ ){
>             $note .= "$_\n" if ($_ ne "\n");
>             print "$_<br>";
>         }
>         else {
>           print "$_<br>";
>         }
593c1634
<
---
>
620c1661,1663
<
---
>     Args::check_bmark();
>     my $bmark = Args::get_bmark();
>
626c1669,1671
<
---
>     # Replace single quote chars w/ html escape sequence. Makes parsing
seq.notes file easier
>     $Args::args{'note'} =~ s/'/&#39/gs;
>
680c1725
<     $str .= "'$1','$Args::args{'note'}'\n";
---
>     $str .= "'$1','$Args::args{'note'}','','$bmark'\n";
689a1735
>       . "Bookmark: $bmark<br><br>\n"
713,714c1759,1760
<         @year, @mon,   @day,  @hour, @min,  @sec, @host,
<         @vol,  @fname, @meta, @data, @type, @note
---
>         @bmark, @year, @mon,   @day,  @hour, @min,  @sec, @host,
>         @vol,  @fname, @meta, @data, @type, @note, @filetype
725c1771
<
/^'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?($::REG_HOST)'?
,'?($::REG_VNAME)?'?,'?(.*?)'?,'?($::REG_META)?'?,'?(\d+)?'?,'?([\w\s]+)'?,'?(.
*?)'?$/
---
>
/^'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?(\d+)'?,'?($::REG_HOST)'?
,'?($::REG_VNAME)?'?,'?(.*?)'?,'?($::REG_META)?'?,'?(\d+)?'?,'?([\w\s]+)'?,'?(.
*?)',?'?(.*?)'?,?'?($::REG_BMARK)?'?\n/
730d1775
<
738c1783,1784
<             $vol[$cnt]   = $8;
```

```
---
>          $vol[$cnt]   = "";
>             $vol[$cnt]   = $8 if (defined $8);
746a1793,1820
>             $filetype[$cnt] = "";
>
>          $bmark[$cnt]   = "";
>          $bmark[$cnt]   = $15 if (defined $15);
>
>             #get filetype if located in seq.notes file
>             if (defined $14 && $14 ne "") {
>               $filetype[$cnt] = $14;
>             }
>             elsif (defined $vol[$cnt] && $vol[$cnt] ne ""
>                    && defined $meta[$cnt] && $meta[$cnt] ne "")
>             {
>               my $ftype = $Caseman::vol2ftype{$vol[$cnt]};
>               my $offset = $Caseman::vol2start{$vol[$cnt]};
>               my $imgtype = $Caseman::vol2itype{$vol[$cnt]};
>               my $img = $Caseman::vol2path{$vol[$cnt]};
>
>               Exec::exec_pipe(*OUT,
>     "'$::TSKDIR/icat' -f $ftype -r  -o $offset -i $imgtype $img $meta[$cnt] |
'$::FILE_EXE' -z -b -"
>               );
>               $filetype[$cnt] = Exec::read_pipe_line(*OUT);
>               close(OUT);
>
>               $filetype[$cnt] = ""
>                 if ( (!defined $filetype[$cnt]) );
>
>               chomp $filetype[$cnt]; #get rid of newline
>             }
768a1843
>           . "<th>Bookmark</th>\n"
771c1846,1847
<           . "<th>Event & Note</th></tr>\n";
---
>           . "<th>Event & Note</th>\n"
>           . "<th> </th</tr>\n";
784a1861,1869
>          # Bookmark
>          if ( (defined $bmark[$i]) && ($bmark[$i] ne "") ) {
>            print "<td align=\"left\" valign=\"top\">\n"
>             .   "$bmark[$i]</td>\n";
>          }
>          else {
>            print "<td> </td>\n";
>          }
>
792c1877
<             # @@@ Why does an error message come up from here:
---
>             # @@@ Why does an error message come up from here:  << Needed to
initialize $vol[cnt] to "" and check if $8 is defined
794c1879,1880
<             if ($fname[$i] ne "") {
---
>             my $dir = "";
>             if ( $fname[$i] ne "") {
805,806c1891
<                     my $fname = "";
```

```
<                     $fname = $1 if ($fname[$i] =~ /^$mnt\/?(.*)$/);
---
>                     $dir = $1 if ($fname[$i] =~ /^$mnt\/?(.*)$/);
811c1896
<                             .
"submod=$::MOD_FILE&vol=$vol[$i]&$Args::baseargs&meta=$meta[$i]&dir=$fname\" "
---
>                             .
"submod=$::MOD_FILE&vol=$vol[$i]&$Args::baseargs&meta=$meta[$i]&dir=$dir\" "
817c1902
<                             .
"view=$File::CONT_FR&vol=$vol[$i]&$Args::baseargs&meta=$meta[$i]&dir=$fname\" "
---
>                             .
"view=$File::CONT_FR&vol=$vol[$i]&$Args::baseargs&meta=$meta[$i]&dir=$dir\" "
847a1933,1934
>
>             print "<br>$filetype[$i]";
852c1939,1969
<             print "<td align=left>$note[$i]</td></tr>\n";
---
>             print "<td align=left>$note[$i]</td>\n";
>
>             # Edit button
>             print "<td align=center>"
>                 . "<a href=\"$::PROGNAME?$Args::baseargs_novol"
>                 . "&mod=$::MOD_NOTES&view=";
>
>             #no vol means it was added manually as a seq event only
>             if ($vol[$i] ne "") {
>                 print "$Notes::EDIT_NOTE&vol=$vol[$i]";
>                 if ($meta[$i] ne "") {
>                     print "&meta=$meta[$i]";
>                     if ($dir ne "") {
>                        print "&dir=$dir";
>                     }
>                 }
>                 #get time that the was originally created.
>                 my $ntime = get_notetime($bmark[$i], $vol[$i], $fname[$i],
$meta[$i]);
>                 print "&ntime=$ntime";
>             }
>             else {
>                 print "$Notes::EDIT_NOTE_MAN";
>             }
>
>             $note[$i] = $1 if ($note[$i] =~ /^\[[MAC]\-Time\](.*)$/);
>             print "&bmark=$bmark[$i]&note=" . Args::url_encode($note[$i]);
>
>             print "\" target=\"_blank\">"
>               . "<img src=\"pict/but_edit.jpg\" alt=\"edit note\" "
>               . "width=45 height=22 border=\"0\">"
>               . "</a></td></tr>\n";
881,882c1998,2014
<       . "<form action=\"$::PROGNAME\" method=\"get\">\n"
<       . "<textarea rows=10 cols=50 wrap=\"virtual\" name=\"note\">"
---
>       . "<form action=\"$::PROGNAME\" method=\"get\">\n";
>
>     my @bookmarks = read_bookmarks();
>     if (scalar @bookmarks > 0) {
>       print "<select name=\"bookmark\" size=\"1\">\n"
```

```
>           . "<option value=\"\" selected>Choose Bookmark</option>\n";
>
>       foreach my $i (@bookmarks) {
>      #make menu of bookmarks
>      print "<option value=\"$i\">$i</option>\n";
>       }
>       print "</select>  \n";
>     }
>     print "New: <input type=\"text\" name=\"bookmarknew\" size=\"15\"
maxlength=\"15\">"
>       . "<br><br\n";
>
>     print "<textarea rows=10 cols=50 wrap=\"virtual\" name=\"note\">"
1007c2139,2182
< # Conver the 'image' format to the 'volume' format
---
> sub get_notetime {
>   my ($bmark, $vol, $fname, $meta) = @_;
>   my $read  = investig_notes_fname();
>   unless (open(NOTES, "<$read")) {
>         return;
>   }
>
>   my $ntime = "";
>   my $skip = 0;
>   while (<NOTES>) {
>     if (/^\-+$/) {
>         $skip = 0;
>     }
>     elsif ($skip eq 1) {
>         next;
>     }
>     elsif (/^\w+\s+\w+\s+\d{1,2}\s+\d\d:\d\d:\d\d\s+\d{4}$/) {
>         $ntime = $_;
>     }
>     elsif (/^Bookmark:\s+(.*)$/) {
>         if ($bmark ne $1) {
>             $skip = 1;
>         }
>     }
>     elsif (/^File:\s+(.*)$/) {
>         if ($fname ne $1) {
>             $skip = 1;
>         }
>     }
>     elsif (/^Volume:\s+(.*)\s\sMeta:\s+(.*)$/) {
>         if ($vol ne $1 || $meta ne $2) {
>             $skip = 1;
>         }
>         else {
>             return $ntime;
>         }
>     }
>   }
>
>   close(NOTES);
>   return "";
> }
>
> # Convert the 'image' format to the 'volume' format
1105a2281,2324
> }
```

```
>
> # Read the bookmarks file and return a sorted list
> sub read_bookmarks {
>     my $fname = bookmarks_fname();
>     open BMARKS, "<$fname" or return;
>
>     my @bookmarks;
>     while (<BMARKS>) {
>         chomp;
>         s/^\s+//;
>         s/\s+$//;
>      push @bookmarks, $1  if (/^($::REG_BMARK)$/o);
>     }
>     close(BMARKS);
>     sort { lc($a) cmp lc($b) } @bookmarks;
> }
>
> # File of bookmark names in list
> sub bookmarks_fname {
>     Args::check_case();
>     $::case_dir = "$::LOCKDIR/" . Args::get_case() . "/";
>     $::case_dir =~ s/\/\//\//g;
>     return "$::case_dir" . "bookmarks.txt";
> }
>
> sub bookmark_add {
>   my $bmark = shift;
>   my @bookmarks = read_bookmarks();
>
>   #check for existing bookmark
>   my $dup = 0;
>   foreach my $i (@bookmarks) {
>     $dup = 1 if ($bmark eq $i);
>   }
>
>   if (! $dup) {
>     Print::log_case_info("Bookmark '$bmark' added to case");
>     my $fname = bookmarks_fname();
>     open BMARKS, ">>$fname" or die "Can't open bookmarks file: $!";
>     print BMARKS "$bmark\n";
>     close(BMARKS);
>   }
>   return;
```

J. PRINT.PM

```
4c4
< # $Date: 2005/09/18 22:49:44 $
---
> # $Date: 2005/09/18 22:49:44 $
79c79
<     print "Content-type: text/html$::HTTP_NL$::HTTP_NL";
---
>     print "Content-type: text/html$::HTTP_NL$::HTTP_NL" unless shift(@_);
108a109,179
> # This routine is used to make the minimum required header statements for the
report pages
> sub print_report_html_header_frameset {
>     my $text = shift;
>     print "Content-type: text/html$::HTTP_NL$::HTTP_NL" unless shift(@_);
>
>     my $time = localtime();
>
>     print <<EOF;
> <html>
> <!-- Autopsy ver. $::VER Forensic Browser -->
> <!-- Page created at: $time -->
> <head>
> <title>$text</title>
> <style type="text/css">
> <!--
>
> body, p, caption, td, li, dd, th  {
>         font-family: Courier New, Courier, mono;
>         font-size: 12px;
>         color: #000000;
> }
>
> .header01 {
>    font-size: 24px;
>    font-weight: bold;
>    background-color: #F5D37A;
> }
>
> .header02 {
>    font-size: 16px;
>    font-weight: bold;
>    height: 30px;
> }
>
> .header03 {
>    font-size: 13px;
>    font-weight: bold;
>    text-indent: 30px;
>    height: 25px;
> }
>
> .header04 {
>    font-size: 12px;
>    font-weight: bold;
>    text-indent: 40px;
>    height 25px;
> }
>
```

```
> .small {
>    font-size: 11px;
> }
>
> -->
> </style>
>
>
> EOF
> }
>
> # Create the header information for the Report Pages
> sub print_report_html_header {
>   print_report_html_header_frameset(@_);
>   print "</head>\n<body bgcolor=\"#CCCC99\">\n";
> }
>
> # Create the header information for the Report Pages Navigation frame
> sub print_report_nav_html_header {
>   print_report_html_header_frameset(@_);
>   print "</head>\n<body bgcolor=\"#f5d37a\">\n";
> }
>
254a326,327
>     #untaint for case reports
>     $fname = $1 if ($fname =~ /^([\w\-\_\/\s\.]+)$/);
256d328
<
260c332
<
---
>
289d360
<
```

K. TIMELINE.PM

```
4c4
< # $Date: 2005/09/18 22:49:44 $
---
> # $Date: 2005/09/18 22:49:44 $
53a54
> $Timeline::BLANK_VIEW = 12;
108a110,112
>         elsif ($view == $Timeline::BLANK_VIEW) {
>             return blank_view();
>         }
466c470
<
---
>                     print localtime() . "\n";
468a473
>                     print localtime() . "\n";
1159,1161c1164,1166
<             if (/^(?:\w\w\w )?(\w\w\w)\s+\d\d\s+(\d\d\d\d)\s+\d\d:\d\d:\d\d/)
{
<                 $url = "tl=$tl_vol&st_mon=$::m2d{$1}&st_year=$2";
<
---
>             if (/^(?:\w\w\w
)?(\w\w\w)\s+(\d\d)\s+(\d\d\d\d)\s+\d\d:\d\d:\d\d/) {
>                 #if no start month show entire month of first entry
>                 $url =
"tl=$tl_vol&st_mon=$::m2d{$1}&st_day=$2&end_day=31&st_year=$3";
1178a1184
>           .
"st_day=$Args::enc_args{'st_day'}&end_day=$Args::enc_args{'end_day'}&"
1185,1187c1191,1199
<       . "<frame src=\"$::PROGNAME?$Args::baseargs&mod=$::MOD_TL&"
<       . "view=$Timeline::VIEW&$url\">\n</frameset>";
<
---
>       . "<frame src=\"$::PROGNAME?$Args::baseargs&mod=$::MOD_TL&";
>
>     if (exists $Args::args{'st_day'}) {
>         print "view=$Timeline::VIEW&$url";
>     }
>     else {
>         print "view=$Timeline::BLANK_VIEW&tl=$Args::args{'tl'}";
>     }
>     print "\">\n</frameset>";
1192a1205,1206
>     Args::check_st_day();
>     Args::check_end_day();
1199,1202c1213,1218
<     my $mon    = Args::get_st_mon();
<     my $year   = Args::get_st_year();
<     my $tl_vol = Args::get_tl();
<     my $tl     = $Caseman::vol2path{$tl_vol};
---
>     my $st_day  = Args::get_st_day();
>     my $end_day = Args::get_end_day();
>     my $mon     = Args::get_st_mon();
>     my $year    = Args::get_st_year();
>     my $tl_vol  = Args::get_tl();
```

```
>     my $tl      = $Caseman::vol2path{$tl_vol};
1218c1234
<       . "<a href=\"$url&st_mon=$pmon&st_year=$pyear\" target=\"_parent\">"
---
>       . "<a href=\"$url&st_mon=$pmon&st_day=1&end_day=31&st_year=$pyear\"
target=\"_parent\">"
1238c1254
<       . "<a href=\"$url&st_mon=$nmon&st_year=$nyear\" target=\"_parent\">"
---
>       . "<a href=\"$url&st_mon=$nmon&st_day=1&end_day=31&st_year=$nyear\"
target=\"_parent\">"
1242c1258
<     # it defaults to the current location
---
>     # it defaults to the current location. User can now specify a start and
end date
1261a1278,1299
>     print "</select></td>\n<td align=\"center\">"
>         . "<select name=\"st_day\" size=\"1\">\n";
>     for my $i (1 .. 31) {
>      if ($i == $st_day) {
>         print "<option selected value=\"$i\">$i</option>\n";
>         }
>         else {
>             print "<option value=\"$i\">$i</option>\n";
>         }
>     }
>
>     print "</select></td>\n<td align=\"center\">"
>         . "<select name=\"end_day\" size=\"1\">\n";
>     for my $i (1 .. 31) {
>      if ($i == $end_day) {
>         print "<option selected value=\"$i\">$i</option>\n";
>         }
>         else {
>             print "<option value=\"$i\">$i</option>\n";
>         }
>     }
>
1293c1331,1333
<       . "Each day that has activity is noted with the number of
events<br>\n";
---
>       . "Each day that has activity is noted with the number of events<br>\n"
>       . "Click the link with the month only to view the timeline for that
entire month<br>"
>       . " or click the link with a given day to view activity on that day
only.";
1302a1343
>         my $day = $a[2];
1309c1350
<               . "<a href=\"$url&st_mon=$mon&st_year=$year\">"
---
>               . "<a
href=\"$url&st_mon=$mon&st_day=$day&end_day=31&st_year=$year\">"
1315,1316c1356,1357
<         print "<tr><td>  </td><td>$a[0]</td>"
<           .
"<td>$a[1]</td><td>$a[2]</td><td>$year</td><td>($a[4])</td></tr>\n";
---
>         print "<tr><td>  <a
href=\"$url&st_mon=$mon&st_day=$day&end_day=$day&st_year=$year\">"
```

```
>             . "$a[0] $a[1] $a[2] $year ($a[4])</a></td></tr>\n";
1328a1370,1371
>     Args::check_st_day();
>     Args::check_end_day();
1333a1377,1378
>     my $st_day  = Args::get_st_day();
>     my $end_day = Args::get_end_day();
1345c1390
<         "$Args::args{'tl'}: Viewing timeline for $::d2m[$st_mon] $st_year");
---
>         "$Args::args{'tl'}: Viewing timeline for $::d2m[$st_mon] $st_day-
$end_day $st_year");
1351a1397,1399
>     # store the date and time
>     my $datetime;
>
1374,1375c1422,1426
<                 if ($date =~ /^(\w\w\w)\s+\d\d\s+(\d\d\d\d)$/) {
<                     if ($2 < $st_year) {
---
>                 if ($date =~ /^(\w\w\w)\s+(\d\d)\s+(\d\d\d\d)$/) {
>                     if ($3 < $st_year) {
>                         next;
>                     }
>                     elsif (($3 == $st_year) && ($::m2d{$1} < $st_mon)) {
1378c1429
<                     elsif (($2 == $st_year) && ($::m2d{$1} < $st_mon)) {
---
>                  elsif (($3 == $st_year) && ($::m2d{$1} == $st_mon) && ($2 <
$st_day)) {
1381c1432
<                     elsif ($2 > $st_year) {
---
>                     elsif ($3 > $st_year) {
1384c1435
<                     elsif (($2 == $st_year) && ($::m2d{$1} > $st_mon)) {
---
>                     elsif (($3 == $st_year) && ($::m2d{$1} > $st_mon)) {
1386a1438,1440
>                 elsif (($3 == $st_year) && ($::m2d{$1} == $st_mon) && ($2 >
$end_day)) {
>                     last;
>                     }
1390a1445,1446
>             #store current date and time for future use
>             $datetime = "$day$date $time";
1395,1404c1451,1480
<
<                 # the deleted meta <blah-dead-2> entries screw up in HTML
<                 $f = "&lt;$1 &gt" if ($f =~ /^<(.*?)>$/);
<
<                 if (($row % 2) == 0) {
<                     print "<tr valign=\"TOP\" bgcolor=\"$::BACK_COLOR\">\n";
<                 }
<                 else {
<                     print
<                       "<tr valign=\"TOP\"
bgcolor=\"$::BACK_COLOR_TABLE\">\n";
---
>             # determine information for creating a note
>             my $url;
>             my $vol;
```

```
>            my $dir;
>            #create hashes of volume numbers and their names and mount points
>            #for easy referencing
>            my %mnt2vol;
>            my %sname2vol;
>            foreach my $i (keys %Caseman::vol2cat) {
>             if ($Caseman::vol2cat{$i} eq "image") {
>               $sname2vol{$Caseman::vol2sname{$i}} = $i;
>             }
>             elsif ($Caseman::vol2cat{$i} eq "part") {
>               $mnt2vol{$Caseman::vol2mnt{$i}} = $i;
>             }
>            }
>
>            #unallocated files match '<imgfile-dead-meta>' or '<imgfile-file-
dead-meta>'
>               if ($f =~ /^\<(.+)-(dead|alive)-(\d+)\>/) {
>              my @imgs_sort_by_len = sort { length($a) cmp length($b) } keys
%sname2vol;
>              my $img = "";
>
>              foreach my $REG_IMG (@imgs_sort_by_len) {
>                 #actually unsafe to match if image and file names contain
dashes. Match known image names.
>               if ($f =~ /^\<($REG_IMG)-(.+)-(\d+)\>/) {
>                 #get rid of lt and gt chars
>                 $dir = "$1-$2-$3";
>                 $img = $sname2vol{$1};
>                 last;
>               }
1405a1482,1504
>              if (exists($Caseman::vol2sname{$img})) {
>                #find the volume the unallocated file lives on by matching mac
times
>                   #This is kind of a shadey hack but I currently don't see a
better way of
>                   #allowing the timeline output to link the meta info to a
volume in Autopsy
>               foreach my $v ( @{ $Caseman::vol2part{$img} }) {         #look
at all partitions until match
>                 if ("$Caseman::tz" ne "") { #set timezone to host's tz
>                   $ENV{TZ} = $Caseman::tz;
>                    POSIX: tzset();
>                 }
>                 my $path2img = $Caseman::vol2path{$img};
>                 my $ftype    = $Caseman::vol2ftype{$v};
>                 my $offset   = $Caseman::vol2start{$v};
>                 my $imgtype  = $Caseman::vol2itype{$img};
>                 my $i_num    = $i;
>
>                 local *OUT;
>                 my $cmd = "'$::TSKDIR/ils' -f $ftype -e -o $offset -i $imgtype
$path2img $i_num";
>                 die "Can't open pipe"
>                   unless defined(my $pid = open(*OUT, "-|"));
>                 if (! $pid) {
>                   $| = 1;
>                    exec("$cmd") or die "Can't exec program: $!";
>                     }
1407,1409c1506,1597
<                  print "<td>$day $date $time</td>"
<                    . "<td>$sz</td><td>$mac</td><td>$p</td>"
```

```
<                   . "<td>$u</td><td>$g</td><td>$i</td><td>$f</td></tr>\n";
---
>                 # Get the fourth line
>                 my $tmp = Exec::read_pipe_line(*OUT);
>                 $tmp = Exec::read_pipe_line(*OUT);
>                 $tmp = Exec::read_pipe_line(*OUT);
>                 $tmp = Exec::read_pipe_line(*OUT);
>                 close(OUT);
>
>                 if ( (defined $tmp) && ($tmp =~
/^$::REG_META\|\w\|\d+\|\d+\|(\d+)\|(\d+)\|(\d+)\|/o) ) {
>                   #get mac time and form to match ils output
>                   my $mtime = localtime($1);
>                   my $atime = localtime($2);
>                   my $ctime = localtime($3);
>                   if ($mtime =~
/^(\w\w\w)\s+(\w\w\w)\s+(\d\d*)\s+(\d\d:\d\d:\d\d)\s+(\d\d\d\d)$/) {
>                     my $day = $3;
>                     $day = "0$day" if (length($day) == 1);
>                     $mtime = "$1 $2 $day $5 $4";
>                   }
>                   if ($atime =~
/^(\w\w\w)\s+(\w\w\w)\s+(\d\d*)\s+(\d\d:\d\d:\d\d)\s+(\d\d\d\d)$/) {
>                     my $day = $3;
>                     $day = "0$day" if (length($day) == 1);
>                     $atime = "$1 $2 $day $5 $4";
>                   }
>                   if ($ctime =~
/^(\w\w\w)\s+(\w\w\w)\s+(\d\d*)\s+(\d\d:\d\d:\d\d)\s+(\d\d\d\d)$/) {
>                     my $day = $3;
>                     $day = "0$day" if (length($day) == 1);
>                     $ctime = "$1 $2 $day $5 $4";
>                   }
>
>                   if ($mac =~ /^m..$/) {
>                     if ($datetime eq $mtime) {
>                      $vol = $v;
>                      last;
>                     }
>                   }
>                   if ($mac =~ /^.a.$/) {
>                     if ("$datetime" eq $atime) {
>                      $vol = $v;
>                      last;
>                     }
>                   }
>                   if ($mac =~ /^..c$/) {
>                     if ("$datetime" eq $ctime) {
>                      $vol = $v;
>                      last;
>                     }
>                   }
>                 }
>               }
>             }
>             #deleted files match 'filename (deleted)'
>             #or symbolic link 'filename -> filename'
>             #or regular filename
>             elsif ( ($f =~ /^(.+)\s\(deleted.*\)/) || ($f =~ /^(.+)\s->\s/) ||
($f =~ /^(.*)$/) ) {
>              $dir = $1;
```

```
>            $dir = $1 if ($dir =~ /^(.+)\s->\s/); #determine filename of form
'fname -> link (deleted)'
>            my $maxlen = 0;
>            foreach my $i (keys %mnt2vol) {
>              if ($dir =~ /($i).*/) {
>                if (length $i > $maxlen) {
>                  $maxlen = length $i;
>                  $vol = $mnt2vol{$i};
>                }
>              }
>            }
>            $dir =~ /^($Caseman::vol2mnt{$vol})(.*)$/;
>            $dir = $2;
>           }
>
>           if (defined $dir && defined $vol) {
>            $url = "&$Args::baseargs&vol=$vol&dir=$dir&meta=$i&recmode=1";
>           }
>
>           # the deleted meta <blah-dead-2> entries screw up in HTML
>           $f = "&lt;$1 &gt" if ($f =~ /^<(.*?)>$/);
>
>           if (($row % 2) == 0) {
>            print "<tr valign=\"TOP\" bgcolor=\"$::BACK_COLOR\">\n";
>           }
>           else {
>            print
>              "<tr valign=\"TOP\" bgcolor=\"$::BACK_COLOR_TABLE\">\n";
>           }
>
>           print "<td nowrap>";
>           if ( ($::USE_NOTES == 1) && (defined $url) ) {
>            print "<a
href=\"$::PROGNAME?mod=$::MOD_NOTES&view=$Notes::ENTER_FILE$url\"
target=\"_blank\">"
>               . "Add Note</a></td>\n";
>           }
>           else {
>            print " </td>\n";
>           }
1411c1599,1610
<                $row++;
---
>           print "<td nowrap>$day $date $time</td>"
>                  . "<td>$sz</td><td>$mac</td><td>$p</td>"
>                  . "<td>$u</td><td>$g</td><td>";
>           if (defined $url) {
>            print "<a
href=\"$::PROGNAME?mod=$::MOD_META&view=$Meta::STATS&$url\"
target=\"_blank\">$i</a></td><td>"
>                . "<a
href=\"$::PROGNAME?mod=$::MOD_FILE&view=$File::CONT_FR&$url\"
target=\"_blank\">$f</a></td></tr>\n";
>           }
>           else {
>            print "$i</td><td>$f</td></tr>\n";
>           }
>
>           $row++;
1413c1612
<         }
---
```

```
>       }
1415c1614
<             print "Error parsing timeline: $_<br>\n";
---
>       print "Error parsing timeline: $_<br>\n";
1417c1616
<     }
---
>       }
1433a1633,1645
>     Print::print_html_footer();
>     return 0;
> }
> # Blank Page for viewing timeline
> sub blank_view {
>     Print::print_html_header("");
>     print "<center><h3>File Activity Timelines</h3>\n"
>       . "Select the range of dates of the timeline you want to view from
above.<br>\n"
>       . "A small range such as a single day is best since this viewer can
be<br>"
>       . "pretty slow with high volumes of data. You may wish to use a
tool<br>"
>       . "like grep on the actual source file found at <br>"
>       . $Caseman::vol2path{$Args::args{'tl'}}
>       . "<br>first to narrow down a time frame.";
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1 Smith, Chris Fred and Bace, Rebecca Gurley, A Guide to Forensic Testimony: The Art and Practice of Presenting Testimony as an Expert Technical Witness," Pearson Education, Inc, 2003.

2 Bogen, A. Chris and Dampier, David A., "Knowledge Discovery and Experience Modeling in Computer Forensic Media Analysis," ACM International Conference Proceeding Series; Vol. 90. June 2004.

3 CERT Coordination Center, "CERT/CC Statistics 1988-2005", http://www.cert.org/stats/cert_stats.html, Last Visited: September 14, 2005.

4 Householder, Allen and Houle, Kevin and Dougherty, Chad, "Computer Attack Trends Challenge Internet Security," IEEE Computer, Vol. 35, No. 4, April 2002, pp. 5-7.

5 Jackson, William, "Digital Forensics Moving from an Art to a Science", Government Computer News, July 29, 2005, http://www.gcn.com/vol1_no1/daily-updates/36556-1.html, Last Visited: September 14, 2005.

6 Anderson, Michael R., "Computer Evidence Processing: Good Documentation is Essential," New Technologies Inc, http://www.forensics-intl.com/art10.html, Last Accessed: September 14, 2005.

7 Palmer, Gary, "A Road Map for Digital Forensic Research," Technical report DTR-T001-0, Digital Forensics Workshop, Utica, New York, 2001.

8 Kruse, Warren G and Heiser, Jay G, Computer Forensics: Incident Response Essentials, Lucent Technologies, 2002.

9 National Institute of Justice, "Electronic Crime Scene Investigation: A Guide for First Responders," July 2001, http://www.ncjrs.org/pdffiles1/nij/187736.pdf, Last Accessed: September 14, 2005.

10 Reith, Mark and Carr, Clint and Gunsch, Gregg, "An Examination of Digital Forensic Models," International Journal of Digital Evidence, Vol. 1, Issue 3, Fall 2002.

11 Carrier, Brian and Spafford, Eugene H., “Getting Physical with the Investigative Process”, International Journal of Digital Evidence, Vol. 2, Issue 2, Fall 2003.

12 Baryamureeba, Venansius and Tushabe, Florence, “The Enhanced Digital Investigation Process Model,” Digital Forensics Workshop, May 27, 2004, http://www.dfrws.org/2004/bios/day1/Tushabe_EIDIP.pdf, Last Accessed: September 14, 2005.

13 Guidance Software, EnCase Enterprise, http://www.guidancesoftware.com/products/ee_index.asp, Last Accessed: September 14, 2005.

14 AccessData, The Forensic Toolkit, http://www.accessdata.com/Product04_Overview.htm?ProductNum=04, Last Accessed: September 14, 2005.

15 Carrier, Brian, The Sleuth Kit version 2.02, July 8, 2005, http://www.sleuthkit.org/sleuthkit/desc.php, Last Accessed: September 14, 2005.

16 Carrier, Brian, The Autopsy Forensic Browser version 2.05, April 8, 2005, http://www.sleuthkit.org/autopsy/desc.php, Last Accessed: September 14, 2005.

17 Carrier, Brian, “Open Source Digital Forensics Tools: The Legal Argument,” @Stake Research Report, October 2002, http://www.atstake.com/research/reports/acrobat/atstake_opensource_forensics.pdf, Last Accessed: September 14, 2005.

18 Carrier, Brian, “Short Bio,” CERIAS – Purdue University, http://homes.cerias.purdue.edu/~carrier/bio.txt, Last Visited: September 20, 2005.

19 Carrier, Brian, “Defining Digital Forensic Examination and Analysis Tools Using Abstraction Layers,” International Journal of Digital Evidence, Vol. 1, Issue 4, Winter 2003.

20 Carrier, Brian, “The Sleuth Kit: Tool Details,” http://www.sleuthkit.org/sleuthkit/tools.php, Last Visited: September 14, 2005.

21 Lee, Rob and Carrier, Brian and Green, John, “Forensic Methodology Illustrated using Linux,” SANS Institute, SANS Fire course books: Track 8 System Forensics, Investigation and Response, Vol. 8.2, 8.3, 2004.

22 Carrier, Brian, “The FAT File System: Sleuth Kit Implementation Notes,” June 2003, http://www.sleuthkit.org/sleuthkit/docs/skins_fat.html, Last Visited: September 20, 2005.

23 NTFS.com, “NTFS Master File Table (MFT),” http://www.ntfs.com/ntfs-mft.htm, Last Visited: September 20, 2005.

24 Burden, Peter, “The Unix File System,” http://www.scit.wlv.ac.uk/~jphb/spos/notes/ufs.inode.html, Last Visited: September 20, 2005.

25 Rivest, Ronald L, “The MD5 Message Digest Algorithm,” Internet RFC 1321, April 1992, http://www.ietf.org/rfc/rfc1321.txt?number=1321, Last Visited: September 18, 2005.

26 National Software Reference Library Project, Reference Data Set, Version 2.9, June 2005, http://www.sleuthkit.org/sleuthkit/index.php, Last Accessed: September 14, 2005.

27 GNU Project, “Grep, Print Lines Matching a Pattern,” January 23, 2002, http://www.gnu.org/software/grep/doc/grep.html#SEC_Top, Last Accessed: September 14, 2005.

28 Mikus, Nick and Korblum, Jesse and Kendall, Kris, Foremost version 1.0, September 09, 2005, http://foremost.sourceforge.net, Last Visited: September 18, 2005.

29 Mikus, Nick, “An Analysis of Disk Carving Techniques,” Master’s Thesis, Naval Postgraduate School, March 2005.

30 Martigonomi, Lorenzo and Bruschi, Danilo and Monga, Mattia, “How to Reuse Knowledge about Forensic Investigations,” Digital Forensics Workshop, 2004, http://www.dfrws.org/2004/bios/day3/D3-Martignoni_Knowledge_reuse.pdf, Last Accessed: September 14, 2005.

31 UNIX Man Pages, “DD(1)”, Last Visited: September 14, 2005.

32 Harbour, Nicholas, DCFLDD version 1.2.4, May 14, 2005, http://dcfldd.sourceforge.net, Last Accessed: September 14, 2005.

33 The Honeynet Project, “The Forensic Challenge”, January 15, 2001, http://www.honeynet.org/challenge, Last Visited: September 17, 2005

34 GNU Project, “Comparing and Merging Files,” http://www.gnu.org/software/diffutils/manual/html_mono/diff.html, Last Visited: September 17, 2005

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Chris Eagle
   Naval Postgraduate School
   Monterey, California

4. George Dinolt
   Naval Postgraduate School
   Monterey, California

5. Cynthia Irvine
   Naval Postgraduate School
   Monterey, California

6. Regis Friend Cassidy
   Naval Postgraduate School
   Monterey, California